Imperial College London

Final Report

Self-Organising Multi-Agent Systems

Department of Electrical and Electronic Engineering
Imperial College London

SOMAS Class 2021–2022

January 2022

Lecturer

Prof. Jeremy Pitt

Abstract

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

Contents

Chapter 1

Introduction

1.1 Problem Statement

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

1.2 Our Goals

Getting a simulation environment working

Chapter 2

Organisation Structure

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

Chapter 3

Simulation Structure

3.1 Simulation Environment

Goes over the backend design

3.2 Simulation Flow

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

3.3 Message Passing

Need a nice diagram for this

3.4 Health Modeling

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

3.4.1 Global Description

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

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

the value of `WeakLevel`. The amount that an agent's HP increases from eating is determined by the function `updateHP()`.

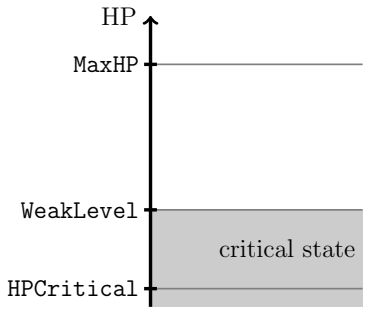


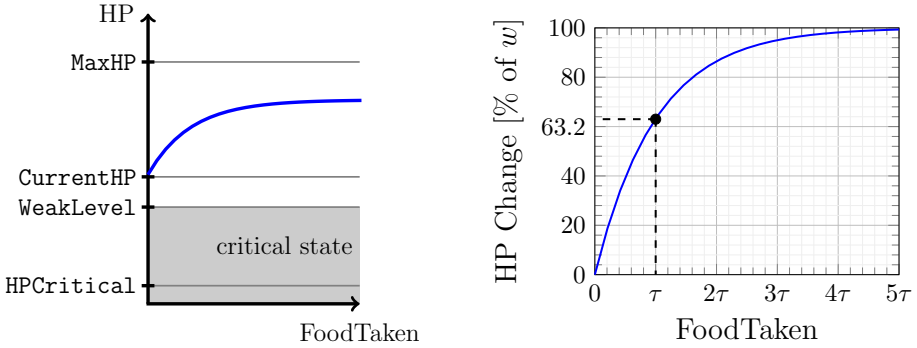
Figure 3.1: The health of the agents is represented by a HP value between `HPCritical` and `MaxHP`. All HP values which are below `WeakLevel` are classed as critical. The diagram is not drawn to scale.

3.4.2 Food and Health: `updateHP`

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

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

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



(a) Overview (b) Detailed representation

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

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

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

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

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

3.4.3 Cost of Living: `hpDecay()`

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

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

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

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

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

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

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

their HP will be set to `WeakLevel`:

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

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

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

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

Chapter 4

Visualisation

Important things about front end.

Chapter 5

Team 2 Agent Design

Chapter 6

Team 3 Agent Design

Chapter 7

Team 4 Agent Design

Chapter 8

Team 5 Agent Design

Chapter 9

Team 6 Agent Design

9.1 Overview

In this chapter, we present the technical formulation and results of the Team 6 agent. This chapter is divided into 3 parts and is structured as follows:

1. Introduce the underlying social motives that our agents can follow in Section 9.3, which define the way our agents utilise common pool resources Section 9.5 and interact with one-another.
2. Explain how the social motives are linked to communication and formation of a social network in Section 9.4.3
3. Illustrate a set of experimental results Section 9.7.

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.

9.2 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, before inheriting the fields contained in table 9.2. 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 9.1: Parameters held in the *Config* data structure

Parameter	Range
BaseAgentRef	*baseAgent
Config	config{} (see Table 9.1)
CurrBehaviour	N
MaxFloorGuess	N
AverageFoodIntake	N
ShortTermMemory	[N]
LongTermMemory	[N]
ProposedTreaties	[*Treaty*]
TrustTeams	map{ *UUID*, N }
Neighbours	{ above:*UUID*, below:*UUID* }

Table 9.2: Parameters held in the *Agent* data structure

9.3 Social Motives

9.3.1 Social Motives

The initial basic concept revolves around the four possible social motives that this agent can have based on different principles, practices, or characteristics. These behaviour types 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 intent to harm the other agents.
- **Narcissism:** An excessive interest or admiration of oneself. A narcissistic agent will act in a way that purely benefits themselves, even to the extent of harming others.

9.3.2 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". One could also think about this in terms of "genotype vs phenotype". For example, an agent may "naturally", "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 decided upon influential factors in changing an agent's social motive and behaviour, include current HP (lower HP means more likely for an agent to act selfishly/narcissistically), and current floor (lower floor means the agent will be worried about food supply, thus may be incentivised to act selfishly).

9.3.3 Willingness to Change

Although the above encapsulates an essential human characteristic, the ability to change, an important element of the human nature, its willingness to change, is missing. Through this next layer of complexity, we attempt to quantify how resistant an agent is to altering their 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 who is initially an altruist, but then may become selfish as the supply of food diminishes, and an agent who is a "true altruist", who will stick to their values for the duration of the game, retaining the same social motive throughout. Similarly, this is extended to the distinction between a true narcissist who never changes, and a narcissist who may, during the game, witness others suffering whilst it itself has an abundance of food, and therefore steer towards a more collectivist behaviour.

9.3.4 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.

9.3.5 Mathematical Formulation

In order to implement our agent's behaviour, we present a mathematical formulation and its implementation for each of the steps introduced above.

Define Social Motives

Social motives can be defined in a spectrum, with one end corresponding to pure altruism and the other end corresponding to pure narcissism. All agents fall on some point on this spectrum

of social motives. 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 more expressiveness. 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 behaviours, and a number closer to 10.0 corresponds to narcissistic behaviour. The boundaries for the 4 distinct social motives are shown in Figure 9.1. The initial "natural" state of a given agent is also shown and is labelled `baseBehaviour`.

The current social motive of the agent is also shown and is labelled `currentBehaviour`. To begin, we set `currentBehaviour` = `baseBehaviour`.

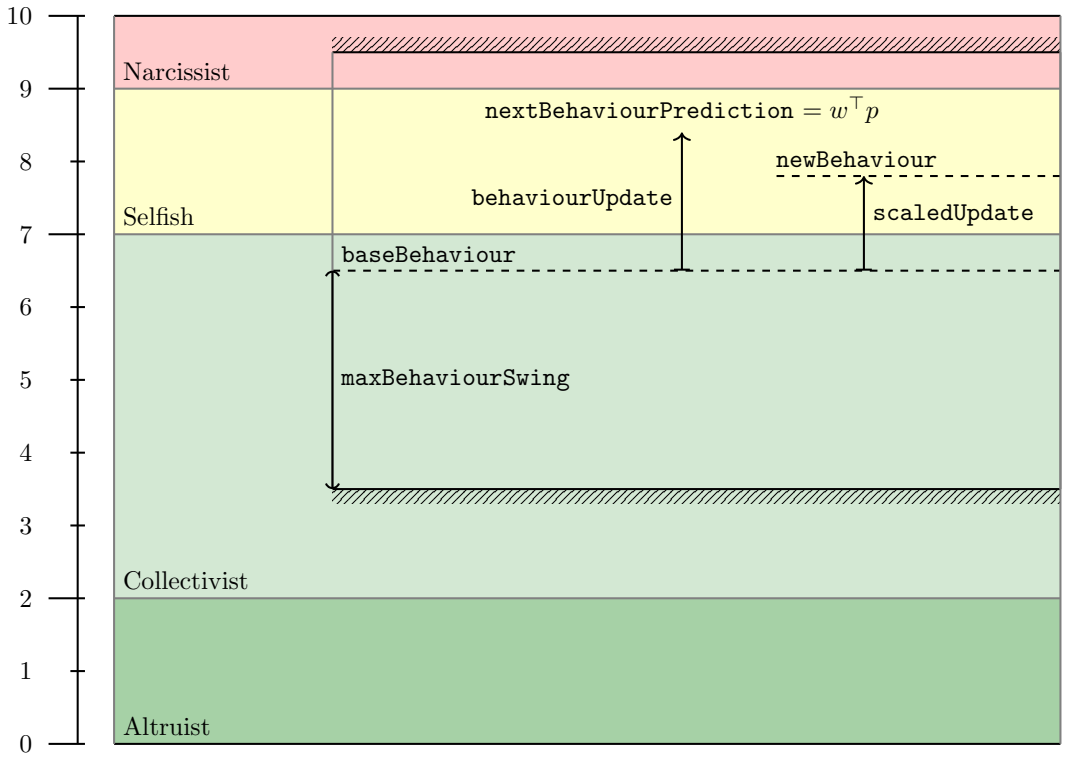


Figure 9.1: Social motive spectrum and update procedure. Add a few additional comments to understand the pictures without having to read to whole text - this is what reviewers typically do at first.

Changing Social Motives

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:

$$\begin{aligned} p &= [currentHP, currentFloor], \quad w = [weightHP, weightFloor] \\ w^\top p &= weightHP * currentHP + weightFloor * currentFloor \end{aligned} \tag{9.1}$$

************* Matt's adaptive weights should be here *************
************* Maybe also HP / Floor utility function *************

Based on how social motive has been mapped between 0.0 and 10.0, a lower floor (*i.e.,* greater floor number) and a lower HP (*i.e.,* smaller HP value) should push the social motive variable to be higher. The weights should then be chosen to reflect these correlations, along with any necessary feature transformations applied to these factors.

This weighted sum represents the prediction of our next social motive, titled `nextBehaviourPrediction` in the Figure below. We therefore calculate:

$$behaviourUpdate = nextBehaviourPrediction - currentBehaviour \tag{9.2}$$

For the first iteration $currentBehaviour = baseBehaviour$ as presented in Fig. 9.1.

Willingness to Change

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 (or, precisely, one minus this number) is multiplied with the behaviour update. This operation is similar to a step-size, and it reflects how easy it is for the social motive to change. We therefore calculate the *scaledUpdate* and *newBehaviour* as shown below.

$$\begin{aligned} scaledUpdate &= behaviourUpdate * (1 - stubbornness) \\ newBehaviour &= currentBehaviour + scaledUpdate \end{aligned} \tag{9.3}$$

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

Bounding Change

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

$$\begin{aligned} max &= baseBehaviour + maxBehaviourSwing \\ \texttt{min} &= \texttt{baseBehaviour} - \texttt{maxBehaviourSwing} \end{aligned} \tag{9.4}$$

If the change as defined in Step 3 above results in a *newBehaviour* outside this range, then this is automatically capped off at the edges. This obviously also holds for the spectrum edges $\{0, 10\}$.

9.4 Trust

9.4.1 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 formation of coalitions in social networks and in determining how high value of information flows through the network" Adali et al. [2010], with Ostrom futhermore supporting that "trust is an essential social lubricant" Ostrom and Walker [2003]. Ostrom further asserts that trust "affects the willingness to cooperate" Ostrom and Walker [2003] and "enhances cooperation in social-dilemma experiments" Ostrom and Walker [2003], 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 Adali et al. [2010] that trust facilitates " communication behaviors which are statistically different from random communications."

9.4.2 Initialisation

The agent takes a simple algorithmic approach to formulating a social network; firstly, the ID of the neighbouring agents is determined through message passing, this ID is mapped to an initial trust value and the trust value 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

9.4.3 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 9.4.4).

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 treaty is evaluated as good: +2, otherwise: -1

Communications

1. If an agent makes a request for another to take 0 food (Section 9.4.3):
   If accepted: reset to 0, otherwise: -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 $\leq$ -10), it will send a message asking them to take 0 food for 1 turn, so as to restore trust and show repentance. If this agent accepts the request, it is forgiven and have their trust value reset to 0. If not, the opinion of this agent continues to decrease (-1).

Case of Low Average Food

If an agent continually averages less that 1 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 decreased (-1). Conversely, if the agent is averaging above this amount, it will increase the trust in the neighbour above (+1).

9.4.4 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 Adali et al. [2010]. 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 trust of another agent will increase as follows:

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

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

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

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

9.5 Food Consumption

9.5.1 Conceptual Description

Now that the agent's behavioural theoretical background has been defined, their corresponding strategy can be defined based on the defining qualities of their social motive. Even though social motives are defined using a continuous spectrum, our agent's consumption and communication strategies were defined on bins along this spectrum, with one bin for each of the four discrete social motives. As social motive goes from Altruist towards Narcissist individual utility is valued increasingly more than collective utility. With regards to strategy, this entails that food consumption increases.

Altruist

Since an altruist is only concerned for the well-being of others, and not concerned about their own well-being, an altruistic agent takes no food and aims to maximise the amount of food left for the remaining agents.

Collectivist

A collectivist will consume just enough food to survive, leaving enough food for the remaining agents. This implies that the first few days the agent will take no food. Later on, when the agent approaches a critical condition, it will eat enough food to reach a threshold value to satisfice (minimally satisfy) themselves.

Selfish

A selfish agent will consume enough food to be satiated and remain in a strong range of health. What constitutes as satiation will be defined subjectively, based on the HP function.

Narcissistic

A narcissistic agent will consume the maximum amount of food possible to take at a given iteration, since it will purely be concerned for its well-being and be willing to harm other agents.

9.5.2 Mathematical Formulation

The implementation for food intake for many agents depends on the health function. Four different strategies are defined for the four defined social motives. Which strategy is carried out is decided in a switch-case statement on the agent's current behaviour.

Altruist

An altruistic agent will always return 0 for food intake. This is reflected in the following case statement:

```
    case "Altruist": // Never eat
        return food.FoodType(0)
```

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 zone, and will die if spending `healthInfo.MaxDayCritical` days in this zone without consuming `healthInfo.HPReqCToW` units of food. The collectivist agent, having entered the Critical zone, will consume food on a randomly assigned day within the allowable period to leave the critical zone. 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 on any given day. This is reflected in the following case statement:

```
case "Collectivist": // Only eat when in critical zone randomly before expiry
    switch {
    case currentHP >= levels.weakLevel:
        a.foodTakeDay = rand.Intn(healthInfo.MaxDayCritical) // Stagger eating days
    return food.FoodType(0)
    case currentHP >= levels.critLevel:
        if a.DaysAtCritical() == a.foodTakeDay {
            return food.FoodType(healthInfo.HPReqCToW)
        }
        return food.FoodType(0)
    default:
        return food.FoodType(0)
    }
```

Selfish

A selfish agent will consume food to remain in a healthy state. A healthy state is arbitrarily defined as the agent's HP being between `0.3 * healthInfo.MaxHP` and `0.6 * healthInfo.MaxHP`. These values are assigned to variables `levels.healthyLevel` and `levels.strongLevel`, respectively. The selfish agent operates under these three conditions:

1. If `currentHP` is greater than or equal to `levels.strongLevel`, the agent takes 0 food because the agent is only concerned of being in the healthy zone.

2. If `currentHP` is between `levels.healthyLevel` and `levels.strongLevel`, the agent takes enough food to maintain their current HP value

3. If `currentHP` is less than `levels.healthyLevel`, the agent takes enough food to reach `levels.healthyLevel`.

Based on the health functions, the change in HP is formulated in equation (9.5):

$$HP^{+} = HP + w(1 - e^{\frac{x}{\tau}}) - (b + s(HP - weakLevel)) \tag{9.5}$$

HP is the agent's current HP, HP^{+} is the updated HP after consuming x units of food. b and

s are the base HP loss and the HP loss slope, defined in the health loss function, respectively. w is the maximum possible HP gain. τ is the food required for the HP to increase by $0.63w$.

Rearranging equation (9.5) to solve for food consumed x, the food required to reach some goal HP from the agent's current HP is formulated in equation (9.6):

$$x_{goal} = \tau ln(w) - \tau ln(w - HP_{goal} + (1 - s)HP - b + s \cdot weakLevel) \tag{9.6}$$

The above equation is implemented in a function that takes `currentHP`, `goalHP`, and `healthInfo` as arguments and returns a `food.FoodType` value in the following manner:

```
func (a *CustomAgent6) foodRequired(goalHP float64, healthInfo *health.HealthInfo) food.Food
    denom := healthInfo.Width - goalHP + (1-healthInfo.HPLossSlope)*a.HP()
            - float64(healthInfo.HPLossBase)
            + healthInfo.HPLossSlope*float64(healthInfo.WeakLevel)
    return food.FoodType(healthInfo.Tau * math.Log(healthInfo.Width/denom))
}
```

This function is sufficient to implement the three conditions of a selfish agent. The selfish agent's strategy is implemented in the following case statement:

```
    case "Selfish": // Stay in Healthy zone
    switch {
        case currentHP >= levels.strongLevel:
            return food.FoodType(0)
        case currentHP >= levels.healthyLevel:
            return a.foodRequired(currentHP, healthInfo)
        default:
            return a.foodRequired(levels.healthyLevel, healthInfo
    }
```

Narcissistic

A narcissistic agent will always return the maximum amount of food consumable:

```
    case "Narcissist": // Eat max intake
        return healthInfo.maxIntake
```

9.6 Self-Organization

"Self-organizing" makes up almost half of this course's name. Accordingly, we approach self-organization capabilities as fundamental to our agents behaviour.

Messages allow for short-lived communication between two neighbours. With request and response messages, such as "leave X food for me", agents can organize locally. However, to establish a stable, self-organizing system across many more floors and many reassignment periods, a more sophisticated form of agreement is necessary. For that reason, an agent's strategies relies 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

Before we explain the theory and implementation answering each question, it is necessary to narrow down what types of treaties could lead to stable systems.

9.6.1 Types of treaties

As food is a scarce good in the tower, the main purpose of self-organization should be to limit how much food an agent eats. While this strategy handles all possible conditions, ("less than", "less than or equal", "equal to", "more than or equal to", "more than"), it focuses on upper bounds to the amount of food we eat. Table X show how different social motives would handle each form of treaty.

Table with motives

The infrastructure of the tower provides multiple types of treaties; "leaveAmountFood", "leavePercentageFood" and "inform". It wasn't clear to me from first read that the next sentences were discussing the problems of each of the treaty types, maybe have to make it more clear? "leaveAmountFood" would allow the first few floors to eat as much as it wants without breaking the treaty. Once the amount is reached, any subsequent agent does not receiving 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 independently of the amount of food an agent receives. 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 treaties are not contradicting one another.

The agents we propose rely on two main underlying principles in order to handle treaties: Forecasting and Utility Theory.

9.6.2 Forecasting

Theory

Treaties do not have any immediate effect. They instead influence the future of an agent. In order to handle treaties today, agents thus greatly benefit from an ability to predict their future.

In the case of this tower, "future" can mean two decisively different things: The future of an agent on the current level and their future on a series of unpredictable levels after that. E.g. an agent very high up in the tower can expect to receive 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 propose each agent separately keeps track of the amount of food it receives each day during the current reassignment period as well as since their first day in the tower. This aligns well with the core assumption in cognitive psychology that there are separate systems for long- and short-term memory.Norris [2017]

Implementation

To implement forecasting, we use two separate arrays corresponding to long-term and short-term memory, respectively and storing the amount of food received each day. The short-term memory is reset after every reshuffling. A scaled histogram measuring the number of times the agent received a certain amount of food results in a discrete distribution predicting the expected food in the future.

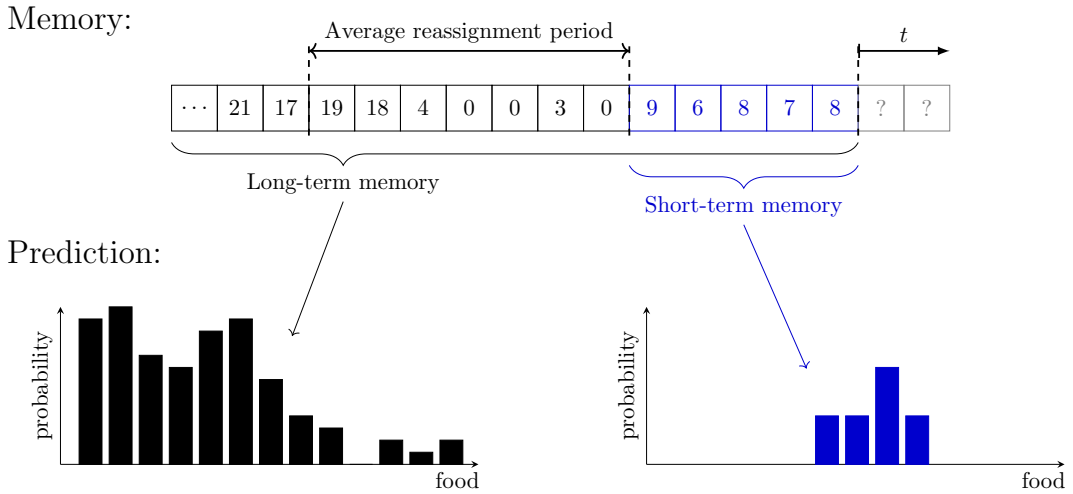


Figure 9.2: Forecasting of the estimated food received in the short-term and long-term using two integer arrays.

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.

9.6.3 Utility Theory

Memory allows this 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. Our agent needs to be able

to compare two potential futures and decide which one it deems more beneficial. Behavioural research has shown that the utility a person gets gets from receiving something is not linear to the value of that thing and differs from. We thus use utility functions in our agents to model human decision making more accurately.Fishburn [1970]

Using utility functions, comparing which one of two different futures is more beneficial is simple. An agent calculates the expected utility with and without a treaty. It then maximise their estimated future benefit by choosing the larger expected utility.

Comparing potential future outcomes with utility allows us to introduce known psychological behaviours to our agents.

The expected utility of a gaining a certain amount of food a is simply $E[U(a)] = U(a)$. The utility of gaining an uncertain amount of food resembled by the discrete random variable x that is based on past experience, is computed given

$$x = [p_1 : x_1, p_2 : x_2; ... \; p_n : x_n] \tag{9.7}$$

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

$$E[U(x)] = p_1 \times U(x_1) + p_2 \times U(x_2) + ... + p_n \times U(x_n) \tag{9.8}$$

Prospect Theory

Prospect theory, first established by Kahneman and Tversky in 1079, is a well established model of how we perceive a change in value, i.e. how much utility we get or loose out of a change in value. Kahneman and Tversky [2013]

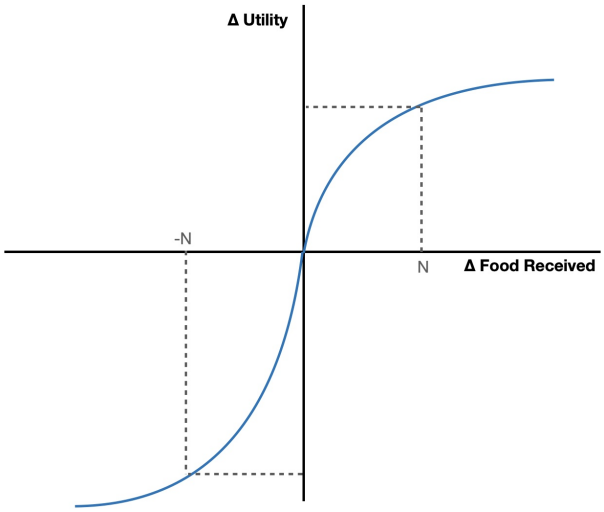


Figure 9.3: Utility in decision making according to prospect theory.

Three main principles make up this model:

- **Greediness:** Humans are generally greedy, i.e. more of something is always at least as beneficial to us. Utility functions hence are generally increasing Fig. 9.3.

- **Diminishing Sensitivity:** Marginal returns are strictly decreasing, i.e. the more we get, the less does one additional utility matter to us. An additional unit of food matters much more to our agents 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 our agent perceives as equivalent to a random distribution, its *certainty equivalent* C, is hence less than its mean.

$$U(C) = E[U(x)] < U(E[x]) \tag{9.9}$$

- **Loss aversion:** Loosing some amount of food is generally perceived as worse than gaining that same amount. Our agents weigh loosing an amount of food stronger than gaining the same amount of food.

9.6.4 Social Motives and Total Utility

Prospect theory forms the basis to how we model our agents benefit from a change in consumption. However, in addition to the *gain,* there is also a *cost* that increases with eating more food: Every unit of food gained is one unit of food other agents lost. This social cost, combined with the previously explained explained utility, forms our total utility function. Each of our social motive corresponds to different values for greediness, risk-aversion, and social cost, which we have chosen according to three insights:

1. The more selfish you are, the more greedy you are.
2. The more you care for the greater good, the higher is your social cost associated with consumption.
3. Studies such as the one by Campbell et al have shown that more narcissistic people are generally less risk-averse. Campbell et al. [2004]

We chose the parameters for each social motive accordingly, with greediness g, risk-seeking r and social responsibility c, resulting in the total utility functions in Section 9.6.4.

- **Altruist**: Not greedy, very risk-averse, very high social responsibility ($g = 0\, r = 1\, c = 5$)
- **Collectivist**: a bit greedy, risk-averse, high social responsibility
- **Selfish**: greedy, little risk-aversion, little social responsibility
- **Narcissist**: very greedy, very little risk-aversion, no social cost

graphs for each utility function, parameters

9.6.5 Weighing the Short- against the Long-Term

As mentioned earlier, the futures our agent should expect in the short run (on the current level) and in the long run differ. The longer the duration of a treaty, the more should the long-term benefit matter. Using the short and long term memory of food received each day, our agent can separately estimate how much he will be getting on the floor he is currently on and how much he will be getting on average across all future reassignment periods.

By calculating the length of average reassignment period, our agents can estimate the number of days remaining on the current floor. We use the proportion of estimated days before the next reshuffling period to the duration of the treaty in order to weigh how much an agent should focus on the short term. Let b_{short} and b_{long} be the estimated long and short 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

$$b_{\text{tot}} = d \times b_{\text{short}} + (1 - d) \times b_{\text{long}} \tag{9.10}$$

where $d = d_{\text{current}}/d_{\text{treaty}}$.

The following examples show how this plays out.

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

Survival Instinct

When one is doing well, the above weighting between the long and short term benefits is sensible. However, if one is struggling to stay alive, survival instincts kick in. Consequently, our agents ignore the expected long-term utility all together when their health is on a critical level.

9.6.6 Implementation

How we rate treaties closely follows the theoretical steps from above. Every day, the agent stores the amount of food received in two arrays, one for the short-term and one for the long-term. The short-term array is reset after every reshuffling period. When a new treaty is proposed, our agent then...

1. calculates what his expected short- and long-term utility is according to Equation (9.8) and using the memory of food received in the past. The utility function used depends on their current social motive.
2. amplifies the utility if it is negative in accordance to mimic loss-aversion.
3. estimates the amount of food he can take under the treaty, i.e. the difference between the average amount of food he received in the past and the amount of much food he needs to leave in order to comply with the treaty, and calculate the utility of taking that amount of food.

4. computes the estimated short-term and long-term benefit of signing the treaty as $U(\text{sign}) - U(\text{don't sign})$.
5. chooses to focus on the long-term or short-term benefit according to Equation (9.10)
6. signs the treaty if it's 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 their total utility function thus needs to be able to vary, too. We accounted for this by introducing a scaling factor z without changing the general properties of the function, leading to:

$$U(x) = g\left(\frac{1}{z}\left(\frac{cr}{g}\right)^{\frac{r}{1-r}} x\right)^{\frac{1}{r}} - cx\frac{1}{z}\left(\frac{cr}{g}\right)^{\frac{r}{1-r}} \tag{9.11}$$

9.6.7 Behaving according to treaties

Treaties are fundamentally built on the assumption that all agents behave truthfully. Introducing dishonesty to treaties would strap them of all their power.

how do experimental results support this?

Accordingly, we made sure that our agents behave in such a way as to never break any accepted treaty.

This also means that our agents will never sign a treaty that contradicts previously accepted bounds on food intake.

9.6.8 Proposing Treaties

Describe how we propose treaties

9.7 Simulation Results and Discussion

9.7.1 Hypothesis

Before running presenting our simulation results, we present the hypothesis we want to verify. Knowing our modeling approach, the following main hypothesis and questions are relevant:

1. In a purely narcissist system, it is impossible for an agent to survive on long-term.
2. In a purely collectivist system, a stable state is reached instantaneously.
3. In a system composed of agents with different social motives, a quasi-stable (define this term) can be reached in a long-term scenario.

   Show oscillation "period", also probably reflecting in the number of deaths per day and the utility per day
4. There exist threshold on the number of selfish agents that can be incorporated in the system before a stable collectivist system becomes unstable.

   Experiment of Wednesday night with TakeFood treaties
5. In a system composed of agents with different social motives, the use of treaties improve the global utility.

   try with vs without treaties on some configuration. for example with the experiment of point 4
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.

   implement genetic replacement

9.7.2 Summary of Simulations

To examine the different hypothesis introduced in Section 9.7.1, different simulations need to be performed. We divide our simulations into n groups, each of them labelled by a capital letter (A, B, etc.) and having a different initialization procedure. Table 9.3 summarizes our simulations, together with their initial conditions.

Genetic Replacement I: 100 Agents, 50 Food, 100 MaxHP, 60 Days

Genetic Replacement II: : 100 Agents, 500 Food, 100 MaxHP, 60 Days

Explain the initialization parameters **@Matt Scott?**

Make the table look good

9.7.3 Simulation Results and Discussion

In this section, we present the results by simulating our system according to Table 9.3.

Simulations A: Single Agent Type

The results are shown in Fig. 9.4.

Name	Behaviour initialization	maxBehaviour Swing	Stubbornness	special feature
A1	Altruist	0	-	
A2	Collectivist	0	-	
A3	Selfish	0	-	
A4	Narcissist	0	-	
B1	Ratio 0.1,0.4,0.4,0.1	0	-	
B2	Ratio 0.55,0.7,0.2,0.05	0	-	
C1	Ratio 0.1,0.4,0.4,0.1	8	0.2	
C2	Ratio 0.1,0.4,0.4,0.1	2	0.8	
D1	Collectivist	6	0.8	LeaveFoodTreaty
D2	Collectivist	6	0.8	TakeFoodTreaty I
D3	Collectivist	6	0.8	TakeFoodTreaty II
E1	Ratio 0.25,0.25,0.25,0.25	6	0.2?	Genetic Replacement I II
E2	Ratio 0.25,0.25,0.25,0.25	4?	0.2?	Genetic Replacement

Table 9.3: Summary of the simulations. Control parameters E1 and E2.

Simulations B: Multiple Agent Types without Behaviour Change

to simulate and add: utility over time, death over time

Simulations C: Multiple Agent Types with Behaviour Change

to simulate and add: utility over time, death over time

Simulations D: Destabilization of a Collectivist System

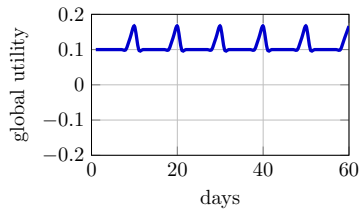
The results are shown in Fig. 9.5.

Add results D3

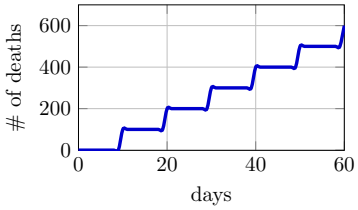
Simulations E: Genetic Replacement of Agents

The results are shown in Fig. 9.6.

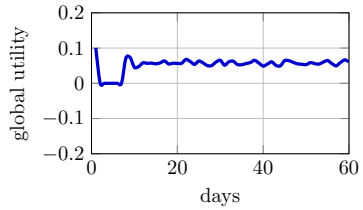
Add results E2



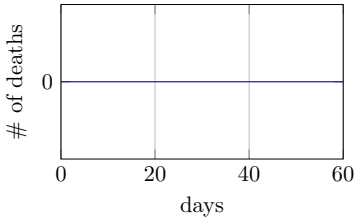
(a) A1: utility over time



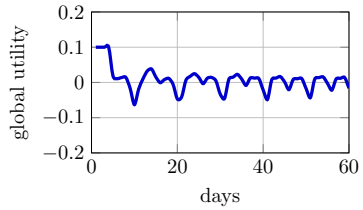
(b) A1: deaths over time



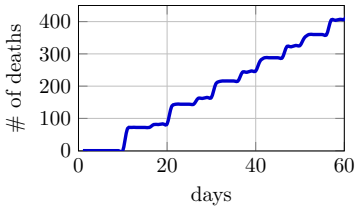
(c) A2: utility over time



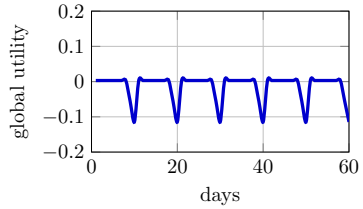
(d) A2: deaths over time



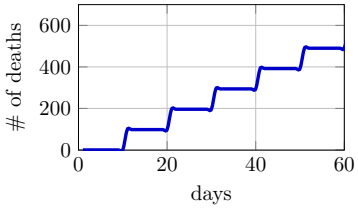
(e) A3: utility over time



(f) A3: deaths over time

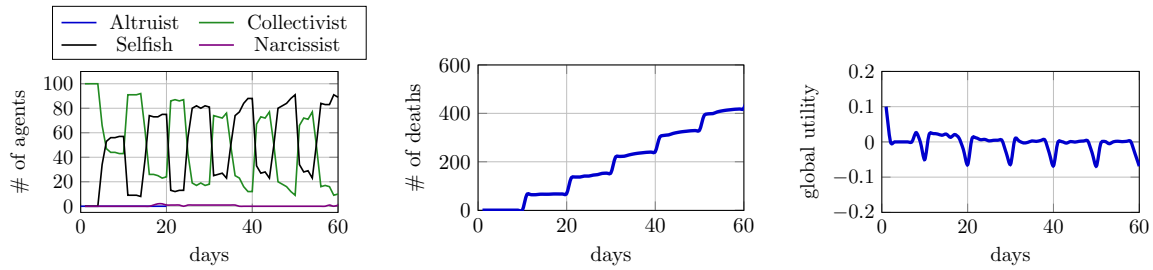


(g) A4: utility over time

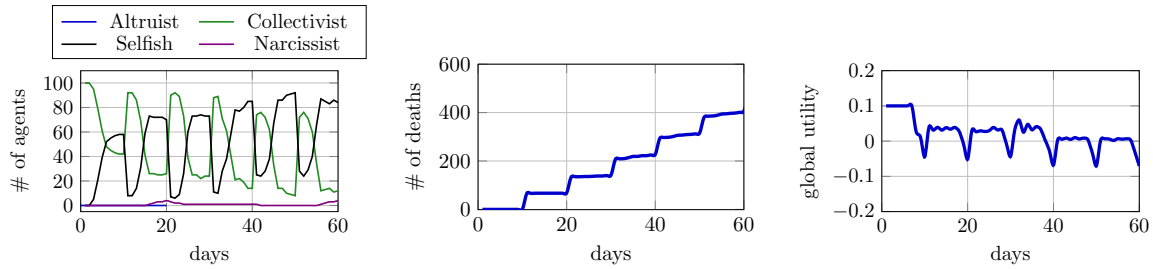


(h) 4: deaths over time

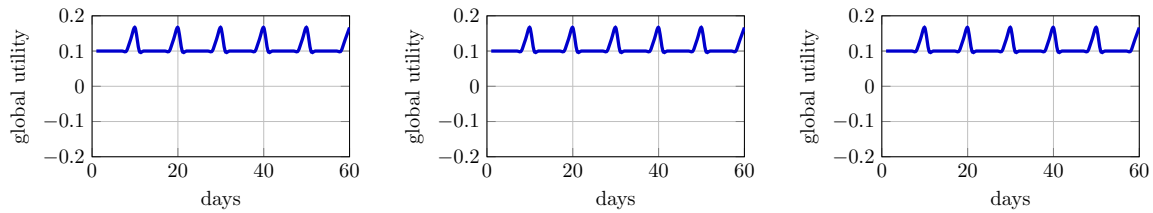
Figure 9.4: Simulation results for a group of agents with uniform social motive.



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

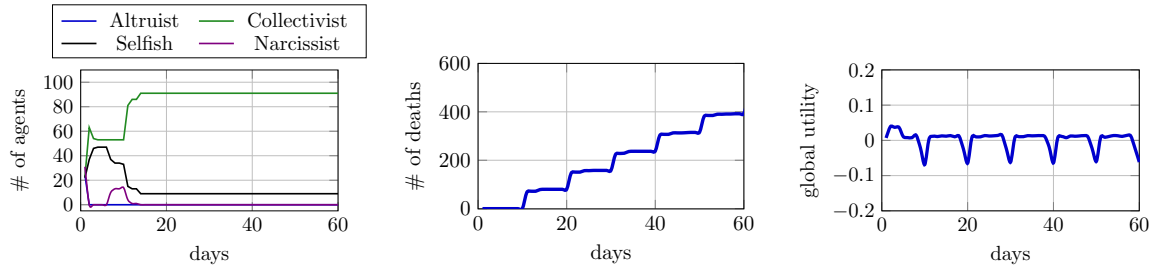


(d) D2: social motives (e) D2: deaths over time (f) D2: utility over time

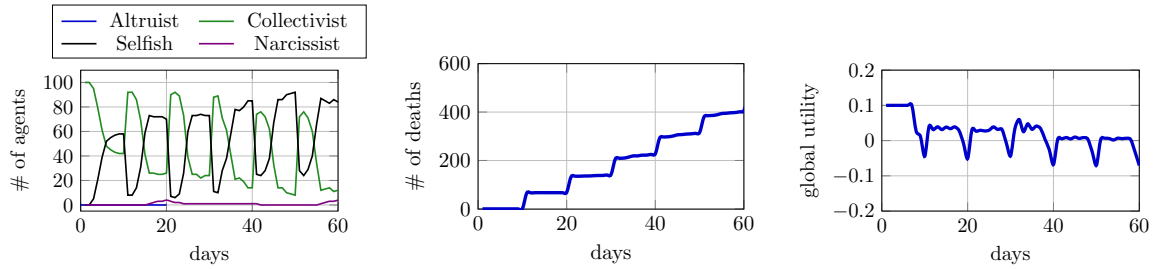


(g) D3: social motives (h) D3: deaths over time (i) D3: utility over time

Figure 9.5: Simulation results different treaties acceptances.



(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 9.6: Simulation results with genetic replacement of the agents.

Chapter 10

Team 7 Agent Design

Chapter 11

Experiments

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

Chapter 12

Results

Chapter 13

Discussion

Chapter 14

Conclusion

Conclusion section.

Chapter 15

Future Work

Futur Work section.

Appendix A

Appendix

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

Bibliography

S. Adali, R. Escriva, M. K. Goldberg, M. Hayvanovych, M. Magdon-Ismail, B. K. Szymanski, W. A. Wallace, and G. Williams, "Measuring behavioral trust in social networks," in *2010 IEEE International Conference on Intelligence and Security Informatics*, 2010, pp. 150–152.

E. Ostrom and J. Walker, *Trust and reciprocity: Interdisciplinary lessons for experimental research*. Russell Sage Foundation, 2003, pp. 24, 209.

D. Norris, "Short-term memory and long-term memory are still different." *Psychological bulletin*, vol. 143, no. 9, p. 992, 2017.

P. C. Fishburn, "Utility theory for decision making," Research analysis corp McLean VA, Tech. Rep., 1970.

D. Kahneman and A. Tversky, "Prospect theory: An analysis of decision under risk," in *Handbook of the fundamentals of financial decision making: Part I*. World Scientific, 2013, pp. 99–127.

W. K. Campbell, A. S. Goodie, and J. D. Foster, "Narcissism, confidence, and risk attitude," *Journal of behavioral decision making*, vol. 17, no. 4, pp. 297–311, 2004.

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