Self-Organising Multi-Agent Systems: Final Report

SOMAS Class 2023

December 15, 2023

Contents

List of Figures

Abstract - In this report, an inter-dependent collective action problem was devised in order to observe phenomena relating to self-organizing multi-agent systems. Altering institutional power through egalitarian, autocratic and totalitarian governance roles, by limiting voting methods and rights. Setting limited accessible information for agents and providing an Agent Common Language (ACL) enforced agents to depend on others for information, thus requiring trust frameworks for agents to be devised. The environment was tuned to adjust the economy of scarcity by limiting collective resources, increasing competition and threat. The ACL showed to improve collective utility, however this improvement was limited due to insufficient messaging implementation for some agents. When leadership changed every 10 rounds, there was a marked decrease in the average lifespan of all agents. The reason for that was the low frequency of changing leadership can cause more totalitarian agents. Longer game simulations and advancements to the agents, particularly in terms of ACL implementation and differing governance strategies are needed for reliable conclusions.

Chapter 1: Introduction

1.1 Abstract

> "The worst part of the punishment is that he who refuses to rule is liable to be ruled by one who is worse than himself."
>
> *Plato's "The Republic," Book VI*[59]

As Plato suggested, it is of utmost importance that all people involve themselves in the commons. Refusing to participate, allows those who do participate to promote their needs to the detriment of the needs of those who do not. In Plato's view, collective participation in the commons is the only way society can flourish. The word "Idiot", derived from the ancient Greek word meaning "a private citizen" [43], was used in ancient Athens to describe people who distanced themselves from the common practices of voting and expressing their opinion. The word's meaning has evolved in modern English to describe a person who is "ignorant" or a "fool", retaining the same underlying meaning that the ancient Athenians intended.

Aligning with this principle, a self-organising multi-agent system is built upon common participation. In SOMAS World, a dynamic electronic environment, multiple fictitious entities termed "agents" venture into a pursuit of survival, energy and point acquisition. These agents confront interdependent collective action challenges amidst varying social relationship scenarios, governed by intricate movement, allocation and interaction strategies. Systems of this nature present interesting implications in computer science and collective intelligence. Independently acting agents equipped with decision-making mechanisms mirror the dynamics of real-world social dilemmas and collective action, offering insight into the capabilities and limitations of such systems. SOMAS World aims to simulate these system's principles. Through observation of the agents, we can address emerging behaviours, decision-making and adapting social dynamics in a test of survival and resource collection.

The Mega-Bike project offers agents a platform to interact, communicate, and stay updated with public broadcasting within SOMAS World. In this environment, each agent-biker develops their own strategies and principles to survive. These strategies are used for the effective control of the Mega-Bike including trajectory planning and contribution to pedalling. Additionally, the agents must consider decisions regarding bike membership and the distribution of energy following its acquisition through "Lootboxes" scattered across the environment. Crucially, agents must cooperate to avoid the entity named "Awdi" which aims to eliminate and remove them from SOMAS World. In reference to self-organizational systems, this project combines the aspects of planning, decision-making, and collaboration to optimize each agent's overall performance.

1.2 Team Structure

To make each team more efficient, the workload was divided, thus allowing each team member to specialize in a domain. Listed below are all the roles and their description.

- **Captain:** The leader of the team, responsible for organization and management of their team. Captains met frequently to discuss the directions that all teams should take to optimize the environment in which the agents acted.
- **Experiments:** Members who conduct experiments on the implemented strategies to draw conclusions on effectiveness of each team's approach.
- **Presentation:** Includes preparation and delivery of the cohort presentation, where each team outlines their strategy and implementation to the cohort.
- **Report:** Responsible for documenting the team's work, preparing text explaining the team's strategies and how it was implemented.
- **Agent Development:** Focuses on code development including, implementing the team's strategies, setting up the software environment.
- **Agent Development:** Developing a platform on which the results of each game were summarized in a visual manner.

All members of each team undertook one of the roles mentioned above, Table 1.1 summarizes how that assignment was done.

Team 1	**Team 2**	**Team 3**	**Team 4**	**Team 5**	**Team 6**	**Team 7**	**Team 8**
Captain							
Lucia Necchi	Ishaan Reni	Xinlong Tan	Xinyi Zheng	Shaheen Amin	Che Ruan	Will Powell	Charlie Chen
Experiments							
Rhea Koury	Alexandra Neagu	Xuanxu Lai	Yanzhou Jin	Arthika Sivathasan	Shijie Wu	Ahmed Safiyanu	Qiyang Yan
Aleera Ewan		Ruiqi Shen			Lijun Chen	Laurie Johnston	Zihao Xue
Rohan Gandhi	Dharmil Shah		Denicke Solomon		Xizhi Chen	Ana Bakoc	Nick Zheng
Presentation							
Chris Myers	Sam Hesketh	Zekai Yushi	Mauro Marino	Arthika Sivathasan	Shutuo Guo	Lia Salvador	Yuhe Zhang
Report							
Timur Ismagilov	Thanasis Kantas	Zhiyu Ma	Divya Ramesh	Omar Zeidan	Yichen Li	Allyson Burba	Kerry Zuo
Agent Development							
Kert Laansalu	Marco Chan	Zhoujing Yuan	Hamed Mohammed	Zack Reeves	Zequan Wen	Dara Costelloe	Charlie Li
Soma Komaromy	Harry Phillips	Tongyu Ding	Lisanth Moodley			Basheq Tarifi	
Sherif Agabiabika	Charmaine Louie	Yuyao He	Panos Doumas	Shaanuka Gunaratnee	Huarun You	Zeyang Ai	Alexander Panagiotidis
Visualisation							
				Shaheen Amin			

Table 1.1: Team Assignments

Chapter 2: Infrastructure

2.1 Game Design

2.1.1 Overview

This report will outline the game structure, detailing the foundational game design and important objectives that guided development. It will start with an overview of the game's intent, exploring how different ideologies of multi-agent systems influence agent interactions within our project framework, as shown in Section 2.1.2. Subsequently, the discussion will shift to the game's aim and provide insights into the design principles that govern agent behavior and environment interaction.

In Section 2.2, the system's architecture part gives details of the various components that build the system. This explains how the multi-agent system is segmented, with each block collaborating to manage agent movements effectively.

Following that, the report discusses the experimental design in Section 2.3 of physics objects in the game. It covers the design aspects of the physical environment, including elements such as the Mega-Bike, Lootbox, and Awdi. Moreover, it describes the physics engine of the project, setting the rules for the agents' movement during the game.

The messaging section 2.4 addresses the agent communication language, detailing how agents communicate and the language format they follow. Based on the structure of our program, this section elaborates on the communication types and format designed, then illustrating the scope of knowledge accessible to agents during this project.

Finally, the Data Collection section 2.5 elaborates on the systematic structures for data gathering. It details various data state structures that facilitate dump processing for usages such as game replays, statistical analysis, and control over simulation visualization. The data collection aspect ensures the consistency of data as well as the fairness of project operation.

2.1.2 Ideologies

The objective for the game design was to establish an inter-dependent collective action problem, that allows for the creation and observation of techniques and phenomena found in self-organizing multi-agent systems. Key components of this system that were devised were resource allocation, consumption, and preservation, in addition to ensuring cooperative game-play and guaranteed self-interest. This was employed in the form of an Ostrom institution[49] framework which involves key aspects of governance, including formulation, policy-making, adjudication management, and appropriation provision monitoring enforcement. Individual decisions impact others, shaping social norms, necessitating fairness in the election of institutional power. This emphasizes the importance of a governance model that promotes equity and shared responsibility in the broader socio-economic context.

The exercise of power within the system is inextricably linked to the concepts of fairness and justice[64]. The satisfaction of agents with their respective leadership, as well as their ability to challenge and replace it, introduces a dynamic equilibrium where power is both granted and constrained by the collective will. In fact, agents always have the possibility to leave the bike they are on, a choice dictated by the desire to strengthen the importance on capability to maintain trust and fidelity for rulers, as ultimately their power is only given scope of expression when they have a following. The parameters governing the dynamics of SOMAS World were in fact designed to heavily penalise lone cycling, as if it didn't that would represent a trivial solution to the problem of solving social conflicts to agree on a goal.

Inspired from Schwartz Theory of Basic Human Values[67], the access for agents' personal information is restricted in the system, such as agents' opinions to others. Instead, the system encourages that agents evolve and increase utility collectively through communication, as discussed in Section 2.4. Voting is also a key component to any self-organising system, and is discussed in Section 2.7. The ideology is to vary the voting mechanisms depending on the governance strategy chosen, to further demonstrate social-cooperation, in addition to the amount of centralized power. The system will aim to allow agents to self-adapt to the fluctuating conditions within the SOMAS environment, mirroring the dynamic nature of cooperative games. In addition, to avoid

the conflicts between agents, voting on the direction resolves disputes between different agents, with the dissenting agents compromising on the agreement resulting from the final vote by applying Alternative Dispute Resolution[54].

The system also employs the Hegelsmann-Krause Model[23], a computational framework for dynamic agent interactions. Agents in the system engage in information exchange and decision-making through message-sharing. This communication influences each agent's opinion of others, and, in turn, guides their subsequent behaviors. The model captures the essence of dynamic decision processes, where agents continually update their beliefs based on received information and observed actions. Through this iterative cycle of communication, decision, and opinion adaptation, SOMAS simulates the complex dynamics of cooperative decision-making among diverse agents.

In this system, all agents manage their own energy assets. The negotiation process for resource exchanges is intricately tied to the pedal force exerted during Lootbox acquisition. This peer-to-peer interaction allows agents to collaboratively share the direction and force of their decisions. The process is monitored by a framework of Regulatory Compliance[33], ensuring that all interactions align with established rules and norms.

During the Founding Institution, detailed in Section 2.1.6, agents are able to choose form into governance with agents who have the same beliefs but there are limits to certain forming groups. To manage Common Pool Resources, we defined ratification processes aligned with the chosen governance model, see Section 2.1.6, ensuring that the system's policies are aligned with its governing structure[74]. This diverse rule endorsement approach fosters a deeper comprehension of cooperative decision-making amidst divergent agent interests and strategies.

Embracing the dual nature of co-dependence and competition, the system requires agents to work together towards communal objectives while simultaneously competing for resources and influence. This dual nature reflects the intricate realities of socio-technical ecosystems[45]. As we explore strategic leadership, power dynamics, and equity maintenance within the system, we will demonstrate how these elements come together in a flexible governance environment where each agent has the capacity to impact the collective direction.

2.1.3 Game Objective

The ultimate goal of the game is for the winning agent to remain alive and accumulate the highest points, primarily through collecting loot boxes of the same color, by the end of the game. Achieving these goal demands each team meticulously devise strategies grounded in game theory, economics principles and sociology insights.

2.1.4 Agent Design

Agents have basic parameters including **onBike status**, **current Mega-Bike ID**. With these parameters, records can be kept for the decisions that each agent performs each round and visualize them.Other parameters such as **energy level** and **points** are gained directly from interactions within the map and reflect their performance, whereby strategies may be compared and evaluated, see Section 13.

To encourage each team to develop strategies to cooperate inter-agent type, agents have a randomly assigned feature, the **soughtColour**, which limits the Lootboxes from which agents can gain points to those of the corresponding colour. Therefore grouping in color dimension and cooperating with agents with various strategy might be a better way to achieve higher score and also to make the SOMAS World dynamics more interesting.

At a very high level, the agents' behaviour must involve being able to give certain response to the SOMAS World in predefined stages. This includes being able to **choose from given governance types** and **vote for leader or dictator** when relevant. They also need to be able to **decide the direction** they intend to go to and how much they will **pedal** for pursuing that goal. After they achieve any loot, they could discuss on how to **allocate resources**. If the cooperation is not that successful, they can also **vote to kickoff** someone, or simply **change to another Mega-Bike**.

Besides that, agents should have more flexible way to communicate with each other and share their opinion before making actual decisions. Agents can **send messages out of given types**, and **handle the messages sent to them** if they believe it is useful and discard others.

We defined various interfaces for each group to implement those behaviors mentioned above. These give all teams a chance to demonstrate their brilliant ideas within a stable and well defined interface, necessary in order to be able to successfully run agents together and to compare their performances.

2.1.5 Environment Parameters

These parameters influence an agent's strategy, affecting whether they prioritize safety, conserve energy, or focus on scoring points. This includes variables defined in `CommonParameters.go`, such as:

- Physics parameters: speed, force, drag, etc., for both agents and the Awdi.
- Resource and energy parameters: points from same-colored Lootboxes, penalties for idleness, collision thresholds, etc.

2.1.6 Simulation

The simulation consists of multiple rounds. Each of them begins with a founding stage in which agents organise themselves onto bikes, followed by a number of iterations in which agents perform actions and experience consequences in the SOMAS World map.

Each round is self contained in the sense that it will be evaluated according to some predetermined metrics (such as life expectancy and average points) in order to be compared with other rounds.

Across rounds agents that died may or may not respawn (depending on experimental parameters) and they will retain memory (expressed in terms of any internal metrics used to characterise a round, such as opinion of other agents and an idea of successfulness) of the previous round.

This way of structuring the game loop was chosen to enable performing experiments tracking the evolution of an agent's behaviour as they build up memory of past events, as well as an opinion of other agents.

The stages composing a round are as follows.

Founding Institution

Agents all start off the bikes. First, they are able to send messages to the other agents, potentially to share what governance type they would like to participate in, or what their opinion is of certain agents (this is particularly relevant in rounds after the first, as agents will have started forming a "memory").

Then agents express what kind of governance method they would like to be used on their bikes. Agents with alike preferences will be put on the same bike. All agents who expressed a preference for the same bike will be spread evenly on the required number of bikes. If the chosen governance requires a ruler (either a leader or a dictator) elections are ran and the winning agent is appointed the role of ruler, which will imply different things according to the governance type. Each governance type is characterised as follows.

N.B. The terminology for each governance type is poorly labelled and is only used to keep consistent with the rest of the report. A more appropriate categorisation for the governance types are Egalitarian democracy, Autocratic Leadership, and Totalitarian respectively.

- Deliberative Democracy: all agents participate in each decision making process through a vote. They all have power to leave the bike, vote towards someone's expulsion, vote on accepting new people on the bike, vote on which direction to pursue and on how to split the Lootbox. Agents engage in a rudimentary form of discussion at each iteration consisting in inbound and outbound flows of information across agents. The agents are required to make an initial proposal for Lootbox direction, before a binding vote is made.
- Leadership Democracy: agents engage in the same voting processes as in the Deliberative Democracy, however for each of these the leader will provide a mapping of weights to be applied to each agent's vote, this grants the leader full autocratic control over voting - ranging from completely silencing one or more agents to giving everyone equal voting power. And it gives the leader power to silence some agents as a form of sanction for disobeying, as well as rewarding very collaborative agents (or any other dynamic envisaged by the specific leader's strategy).
- Dictatorship: all the decisions that are to be made in a iteration are made by the dictator. Agents don't participate in any form of voting and their only tool to express dissidence is the steering wheel: they can interfere with the dictator's steering. However as the dictator is the only agent with power to kick people out the bike this could incur in a penalty, depending on the dictator's strategy.

The election process for the ruler (where relevant) and a more thorough description of their powers is as follows (whereas the modality through each of the other processes mentioned above happens will be discussed in the relevant sections):

Vote For Ruler

- **Leadership Democracy:** The elected leader will be the ruler until one of the following happens:
  - The leader is kicked out the bike
  - The leader decides to leave the bike
  - The leader dies
  - The round ends

  In all scenarios a part from the latter new elections are held if there are any agents left on the bike. Agents are allowed to leave the bike and they participate in all the voting processes.
- **Dictatorship:** The dictator will remain in power until one of the following events occurs:
  - The dictator leaves the bike
  - The dictator dies
  - The round ends

  As for the leadership case new elections happen if there are still bikers on board (and if it's not the third scenario.

Iteration Loop

In the world of self-organizing multi-agent systems that the SOMAS framework presents, the iteration loop stages are designed to perform the teleo-reactive program[46], which includes norms, strategic decision-making, and governance structures. They dictate the interactions between each agent and orchestrate the collective management of shared resources, all aimed at achieving the shared objectives. The iteration loop is the core component of a simulation, with agents making decisions regarding targets, whether to stay or leave bikes, and managing resources.

1. **STAGE 1: Agents Not on Bike Decide Target**
   All agents that have left/ have been kicked out in the previous round or before and are now in Limbo, and as such off a bike, decide which bike they will try and join. This decision can depend on various factors, including previous results obtained in trying to join a bike, personal opinions of players or governance types, etc.
2. **STAGE 2: Decide to Stay or Leave**
   Agents that at the beginning of the round are on a bike can decide whether they would like to leave the bike in the current round or to participate in the movement.
   If an agent decides they want to leave their current bike the Server will remove them from the current one and place the agent in the limbo, where they will incur in an idleness related penalty, dictated by **LimboEnergyPenalty**.
   For one whole round the agents will be in the limbo. They will be able to try and join a new bike at the beginning of the next iteration. If they are not accepted by their target bike they will have to remain in limbo for another round, losing energy and not being able to loot the boxes. This dynamic encapsulated a dis-incentive from hopping from bike to bike as it poses a risk for the agents, especially as they start building an opinion system.
3. **STAGE 3: Decide on Kick Out**
   In this stage agents may be kicked out from the bike as a punishment for anti-social or non-collaborative behaviour or as a consequence of their strategy. The way the kick out process happens depends on the governance type.
   - On a **Dictatorship** governed bike, the dictator only decides on which agent (if any) will be kicked out. Through this tool the dictator can exercise power over the biker on their bike and potentially punish them for interfering with the steering.
   - On a **Deliberative Democracy** bike a vote will be held to decide on kick out. If an agent receives a number equal or bigger than half the bikers on the bike of votes supporting kick out they will be expelled.
   - On a **Leadership Democracy** governed bike the voting works in the same way as the Deliberative Democracy but the votes will be weighted according to the leader's will.

4. **STAGE 3: Accept Those Who Wish to Join the Bike**

   - On a **Dictatorship** governed bike, agents do not vote on who to accept. This decision is made by the dictator only.
   - On a **Deliberative Democracy** governed bike: Only if there are spare seats, all agents on a bike check bikers who have requested to join. Agents will then express their preference on accepting each of the requesting bikers. Requesting agents with a number of favour votes bigger than half the bikers on the bike will be accepted. If there are more requesting agents than spaces on the bike agents will be sorted based on the number of votes in favour and accepted sequentially until all empty spaces are filled.
   - On a **Leadership Democracy** governed bike the process is analog to that carried out on the Deliberative Democracy one, but final votes are weighted according to the leader's will.

5. **STAGE 4: Direction Choice**
   On a **Dictatorship** governed bike agents do not participate in the decision for a direction, they will be communicated which direction the dictator has chosen when it's time to decide on the pedalling and steering forces
   On a **Deliberative Democracy** and **Leadership Democracy** governed bikes :

   - Agents make their proposal for the direction for next round by expressing a preference for a Lootbox (or abstaining from voting to signify they want to run away from the Awdi instead)
   - Agents formulate a vote for each of the proposed Lootboxes, this will look like a map from each Lootbox to a float64 between 0 and 1. all the votes need to sum up to 1
   - The winning direction is selected using each agent's distribution and one of a number of possible voting methods, which can be varied in experiments (ie majority, ...). In the case of a Leadership the final vote will be affected by the leader's weights

6. **STAGE 5: Decide Pedal and Steer Actions**
   Agents decide on their actions internally regardless of the government type in each government .
   Agent Output a struct containing the following:

   - Pedal Value (Between 0.0 and 1.0)
   - TurningDecision struct containing:
     - a boolean variable indicating whether an agent desires to steer or not (TurningDecision.SteerBike)
     - and a floating point number between -1.0 and 1.0 indicating the desired change in angle with respect to the current bike orientation (TurningDecision.TurningForce)
   - Braking Value (Between 0.0 and 1.0)

   When the governance method is a **Dictatorship** agents should abstain from providing a steering force as only the ruler will be submitting that. However they still have the power to sabotage the ruler's decision by submitting a force regardless.

7. **STAGE 6: Vote Resource Allocation**
   On a **Dictatorship** governed bike the dictator decides on the allocation
   On a **Deliberative Democracy** and **Leadership Democracy** governed bikes :

   - Each agent makes a proposal for the distribution of the loot by associating to each agent on the bike a number between 0.0 and 1.0 corresponding to which percentage of the loot should be allocated to each biker. The distributions are collated and the loot is distributed according to the cumulative one (weighted by the leader in the case of a Leadership)

2.2 System Architecture

In this section, we delve into the system architecture as shown in Figure2.1 which is inspired by PreSage-2 platform[37], outlining the various components that collectively create a dynamic, interactive gaming environment. Each component plays an important role in shaping the game's landscape.

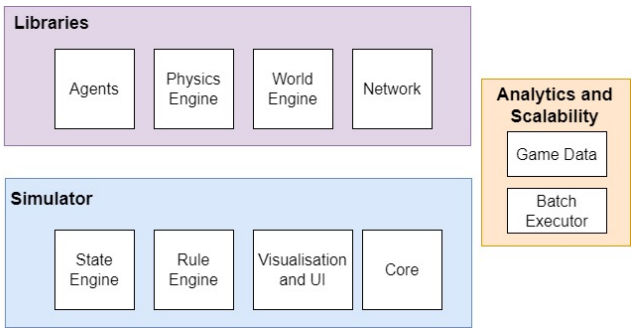


Figure 2.1: System Architecture

Core

Core is essential for ensuring that the game operates smoothly and consistently, maintaining the integrity of the simulation and providing a stable platform for the more specialized systems and processes to function effectively. This is achieved through the modularization of a game and it's division into unitary and repeatable tassels differentiated by the length (in unit time) that they wrap (going from the smallest tassel of one time unit: the iteration, to bigger tassels of hundreds of time units: the rounds). In terms of general functionalities and guarantees it includes:

- Overseeing the progression of the game through various stages or cycles (games, rounds and iterations as presented in Section2.1.6).
- Managing the rules and mechanics of the game environment.
- Coordinating interactions between agents and the game world through the simulation loop and the round loop.
- Handling core game logic, such as the application of game rules and the resolution of game events.

State Engine

State engine ensures that the game's world is accurately represented at all times, reflecting the ongoing interactions and transformations within the game. It includes:

- Tracking the current status of all game elements, such as agent positions, energy levels, and resources.
- Recording changes in the game world, like the appearance of Lootboxes or changes in Mega-Bike states.
- Maintaining historical data for analysis and decision-making processes.
- Updating the game state in response to agent actions and environmental changes.

The maintenance and distribution of an up-to-date game state in fundamental both to guarantee an accurate reproduction of the game through the visualisation engine (which relies on a time-series of game-states), and to ensure that agents are enacting their strategy based on the most up to date game-state.

Rule Engine

By ensuring that all events and activities within the game follow its established rules and logic, the rule engine helps to preserve the integrity and consistency of the game. It includes:

- Evaluating actions of agents against the established game rules and mechanics (for example, ensuring a process consistent with the current bike's government type is followed).
- Determining the outcomes of these actions based on the rules.
- Ensuring that all agent interactions comply with the game's framework: the rule engine will guard against illegal actions, such as targeting a non-existent Lootbox.
- Integrating with an external component (like Awdi) to manage rule applications in specific scenarios, such as that of a deadly collision with the Awdi.

Libraries

Libraries serve as a resource for developers to efficiently implement diverse game scenarios and agent behaviors, ensuring a versatile and dynamic game world.

- The environment library includes the environment parameters (described in Section2.1.5) and physics of the game (described in Section2.3).
- The agent library includes the scripts and algorithms for agent decision-making processes, tools for customizing and extending agent functionalities, and resources for simulating diverse agent interactions and strategies.

Network

Network is used for creating a realistic and functional communication system within the game, essential for coordinating agent interactions and strategies. It includes:

- Simulating the transmission and reception of messages between agents.
- Handling the complexities and constraints of in-game communication networks.
- Allowing for the testing and implementation of various communication strategies and protocols.

More detailed information of the workings of the communications network are provided in Section2.4

Database

Database is for analysing game dynamics, understanding agent behaviors, and refining the game based on empirical data. It includes:

- Recording detailed logs of each simulation run, including agent actions, environmental changes, and game outcomes.
- Storing historical data for analysis, such as agent performance metrics, decision-making patterns, and game evolution.

Batch Executor

Batch executor is a tool for exploring the dynamics of the game under different conditions and for systematically measuring the impact of various independent and dependent variables in the game's environment and agent behaviors. It includes:

- Enabling the execution of numerous simulation runs with varying parameters.
- Collecting data across these multiple runs for comprehensive analysis.

2.2.1 Game Flow Diagram

2.3 Physics in SOMAS World

2.3.1 Background

In SOMAS World, the absence of physical boundaries allows bikers the freedom to explore without limitations. However, a strategic element is introduced by confining the spawning of loot boxes to a specific area within the expansive map.

Physics Boundaries

Unlike traditional games environment with rigid physical boundaries, this gaming world embraces a limitless expanse. Players are not constrained by invisible walls, encouraging exploration and discovery. Venturing beyond the usual confines of a game world is not penalized, but pedalling the Mega-Bike would consume energy while energy can only be restored by collision with Lootboxes. Agents must carefully manage their energy reserves, otherwise they could die by exhausting all the energy.

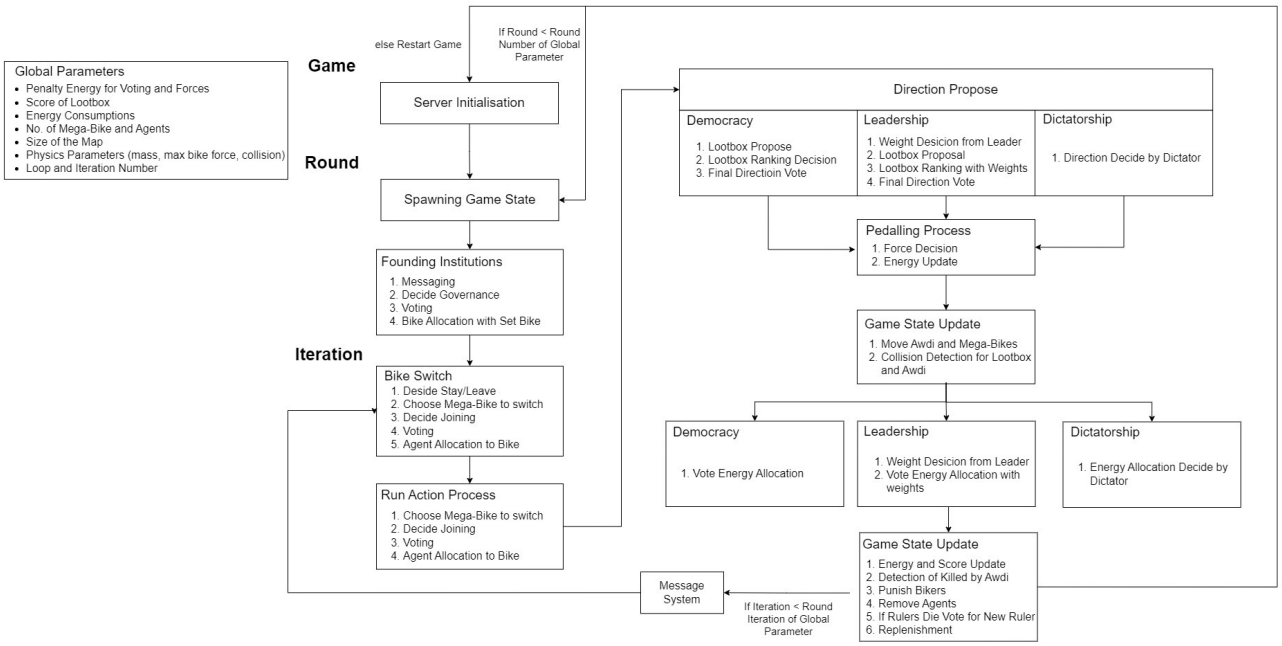


Figure 2.2: Game Flow diagram, illustrating the procedure on a game, round, and iteration level.

Lootbox Spawning Area

Despite the absence of physical barriers, Lootboxes exclusively manifest within a designated region of the map. This region is defined by the dimensions of a grid, specifically a square area with a width and length of 250.0 units each. This deliberate constraint introduces a tactical aspect, as players are motivated to explore and engage within the specified Lootbox spawning zone.

The grid dimensions, set at 250.0 units by 250.0 units, ensure that the Lootbox distribution remains concentrated, creating hotspots of activity while leaving other areas untouched. This dynamic distribution strategy enhances the game-play experience by concentrating valuable resources in specific regions, prompting players to strategically navigate and compete within the designated Lootbox spawning area.

2.3.2 Lootbox

Lootboxes are mainly characterised by:

- Physical: Lootbox uuid; Coordinates on the map
- Color: Their colour (which determines which agent will receive some points from looting a specific Lootbox)
- Total loot: Their energetic quantity (this may vary from Lootbox to Lootbox in some experiments or be fixed in others)

When a bike ends in a position within a certain radius (7.0 units) of the position of the Lootbox, the box is considered looted and bikers are allocated resources according to internal repartitions and points according to the agent's sought colour and the Lootbox's colour.

After a Lootbox is looted it will disappear from the map. If two bikes loot the same box in the same iteration the total amount of energy present in the Lootbox will be equally split between the two bikes. The Lootboxes may or may not be replenished at the end of an iteration, depending on experimental design.

2.3.3 Awdi

In order to motivate agents to move around the map strategically and maintain a high enough average speed (instead of just remaining stationary and waiting for Lootboxes to spawn near them, an antagonist element is introduced: the Awdi.

The basic goal of Awdi is to move towards the slowest/stationary Mega-Bike and remove all agents on it as a punishment. In the game loop, Awdi will follow three steps: receive new information on the current state of the game, update the target Mega-Bike, pedal towards target at a certain (fixed) force.

There are three parameters can be tuned to vary the behavior of the Awdi during experiments:

- 'AwdiTargetsEmptyMegaBike' bool: If true, Awdi will choose target from all MegaBikes. If false, Awdi will ignore MegaBikes with no agent on them.
- 'AwdiOnlyTargetsStationaryMegaBike' bool: If true, Awdi will only target stationary Mega-Bikes and ignore moving ones, when all Mage-Bikes in the SOMAS World is moving then Awdi will stop pedalling. If false, Awdi will find the slowest Mega-Bike as target.
- 'AwdiRemovesMegaBike' bool: If true, Awdi will remove Mega-Bikes when colliding with them. If false, Awdi will only remove agents on it and leave an empty Mega-Bike there.

2.3.4 Mega-Bike

Mega-Bikes objects are what allow agents to actually move around in the map and perform actions. Since agents don't have a location attribute themselves (as when they are not on a bike they aren't technically on the map), they need to be associated to a Mega-Bike in order to move and Lootboxes.

Mega-Bikes store information related to the current governance, such as the type of governance and the id of the ruler (when relevant). This information is used in various action processes, such as to retrieve the leader's weights on votes or to get the dictator's decision on direction (among several other processes better described above).

Furthermore, information on which biker is present on a bike at any iteration is available through the Mega-Bike object.

Structural Features of Mega-Bike

The Mega-Bike is characterized by various structural features that define its properties and dynamics within SOMAS World:

- **Physical:**
  - Mega Bike UUID
  - Coordinates on the Map
  - Total Mass of Mega-Bike (Mega-Bike Mass + Total Mass of Agents)
  - Acceleration
  - Velocity
  - Orientation
  - Resultant Force from All Agents
- **Agent List:** An array of BaseBiker objects representing all agents on the Mega-Bike
- **KickedOutCount:** An integer representing the number of agents to be kicked from the Megabike in the next iteration
- **Governance:** Decision of Governance for Mega-Bike: Democracy; Leadership; Dictatorship
- **Ruler:** Agent UUID of the ruler on this Mega-Bike

These features collectively contribute to the overall behavior and decision-making processes of the Mega-Bike, shaping its role and interactions within the dynamic environment of SOMAS World.

2.3.5 Physics Engine

The Physics Engine is responsible for managing all physical computations related to coordinate updates and collision detection within the context of GameState updates.

- **Important Physical Parameters:**
  - Collision Threshold: The threshold distance for detecting collisions (used in Awdi and Lootbox collision detection). Value = 7.0
  - Mass of Mega-Bike: 1.0
  - Mass of Single Biker: 1.0
  - Mass of Awdi: 7.0

- Drag Coefficient: 0.5 (coefficients for drag force resisting Mega-Bike movement)
- Limbo Energy Penalty: -0.05 (energy lost per round if the agent is off the Mega-Bike)

- **Updating Stage for Mega-Bike:**
  1. Agents decide pedalling/braking/steering force.
  2. Take the average for steering force and update Mega-Bike Orientation.
  3. Sum up the pedalling force and subtract the braking force to decide the resultant forward force in the chosen direction.
  4. Compute acceleration resulting from the cumulative force.
  5. Update the velocity of Mega-Bike (velocity always greater or equal to zero).
  6. Update the coordinates of Mega-Bike based on orientation and velocity.
- **Physics Formulas Used for Computation:**
  - Drag Force:

$$f_d = \lambda_c \cdot V^2$$

    where f_d is the drag force,λ_c is the drag coefficient and V is the current velocity of Mega-Bike.

  - Orientation Updating

$$F_s = \sum_{i=1}^{n} f_{si}$$
$$\hat{f_s} = \frac{F_s}{n}$$
$$\theta_t = \theta_c + \hat{f_s}$$

    where $\hat{f_S}$ represent the average steering force applied by agents who haven't abstained while F_s is the cumulative of steering force and n is the total number of agents applying steering force. The new orientation θ_t is updated from current orientation θ_c by adding the average steering force.

  - Acceleration Computation:

$$F_p = \sum_{i=1}^{M} f_{pi}$$

$$a = \frac{F_p - f_d}{M}$$

    where F_p represents the cumulative result of pedalling and braking applied by agents on the Mega-Bike while M is the number of agents on the bike, f_d is the drag force caused by velocity. And a is the resultant acceleration of Mega-Bike on the decided orientation.

  - Velocity Updating:

$$V' = V + a$$

    where V' represents the new velocity of Mega-Bike while V is the current velocity of Mega-Bike and a is the computed acceleration.

  - Position Updating:

$$\text{bike.X'} = \text{bike.X} + V' \times \cos\theta_t$$
$$\text{bike.Y'} = \text{bike.Y} + V' \times \sin\theta_t$$

    The new Mega-Bike coordinates $(bike.X', bike.Y')$ on the map is updated from the current Mega-Bike coordinates $(bike.X, bike.Y)$. And V' is the updated velocity while θ_t represents the updated orientation of the bike.

2.4 Messaging & Scope of Agent's Knowledge

In the core of our project lies the essential element of communication and information exchange among agents. This necessitates the development of an agent communication language, enabling straightforward yet meaningful interactions within the simulated environment. The information exchange is vital for agents to gather knowledge and make informed decisions regarding various aspects like social rankings or utility scores.

Through the usage of the Base SOMAS package messaging infrastructure agents have the capability to broadcast specific pieces of information to a selected group of agents. In the same way they are able to handle incoming messages from other agents.

This exchange is transparent; any agent listed as a recipient can access the information directed at them. However, it's crucial to note that while constructing and sending these messages, agents have the option to either tell the truth or deceive, depending on their group's strategic approach. Consequently, when receiving information, each group must consider the possibility of the information being inaccurate or misleading, as agents might not always be truthful.

The types of information accessible in the game loop through messaging include:

1. **Reputation Perception**: Agents can acquire other agents' perceptions of a specific agent's reputation, providing insights into their views about the target agent.
2. **Kickout Intentions**: Information regarding an agent's desire to kick another agent out of the current Mega-Bike.
3. **Agreement on Joining Requests**: Understanding whether the information sender agrees with the current agent's request to join a Mega-Bike.
4. **Governance Preference**: Insights into the sender's preferred governance type for the Mega-Bike.
5. **Pedal Forces Information**: Details about the specific agent's applied pedal forces.
6. **Voting Results on Governance**: Information on how the agent voted regarding the governance type.
7. **Voting Results on Leadership**: Insights into the agent's voting choice for a leader.
8. **Voting Intentions for Kickouts**: Understanding the agent's vote regarding which agent they wish to kick out.
9. **Lootbox ID Acquisition**: Ability to acquire the Lootbox ID that the information sender desires.
10. **Lootbox Preference Voting**: Information on how the agent voted regarding different Lootbox preferences.

Agent Communication Language

In order to be able to quickly and easily extend the language's scope a different message type was implemented for each different type of information to be sent. However all messages retained the same overall structure for ease of readability and to increase modularity. The SOMAS Base Platform package was used to implement the sending and receiving of messages.

Creating messages follows the same types as receiving information. Each group's agent will formulate their strategy to communicate the information they wish to send to the server, which will then be accessible to other agents.

In this environment, communication is not just about the exchange of information; it's about strategy, decision-making, and the potential for misinformation, all of which play a crucial role in the dynamic interplay of agents within the simulation.

1. **Structure of ACL**: The ACL is designed to allow for a wide range of information types to be communicated, so the structure of this needs to be in a common structure. In the ACL messaging structure, each message has a fixed structure comprising the following parts: 1. Sender ID information, 2. Intended receivers, 3. Message type (refer to the actual message type mentioned above), 4. The message content itself. The message structures defined in this project are based on the current communication needs between agents and can be further expanded with the increase of communication requirements by adding new types of messages within the existing ACL framework.
2. **Syntax and Semantics**: The ACL uses a standardized syntax and semantics to ensure that messages are consistently interpreted by all agents. For instance, in actual communication messages, the opinion

value corresponding to the sender's and receivers' IDs should all be maintained as float numbers between 0 and 1 and must associated with the corresponding agent ID.

3. **Strategic Communication**: In this environment, communication is a tool for strategic advantage, and through messaging, it's possible to communicate different desired governance types, to identify the agent ID that wants to kick out from the current Mega-Bike, or to agree or disagree with an agent's request to join Mega-Bike, etc. So each group can set their strategies based on the communication information.
4. **Interoperability and Integration**: The ACL ensures the effectiveness of communication among different agents within the current project. Through this, different agents are able to interact with other entities, turning the entire project into an whole with information transmission. It is a critical element in strategy formulation, decision-making, and handling the strategies based on agents interactions.

2.5 Data Collection

The server contains a series of structures known as "Dumps" (`GameStateDump`, `AgentDump`, `BikeDump`, etc) that correspond to every entity in SOMASWorld. A `GameStateDump` contains the entire state of SOMASWorld (expressed as collections of other dump types) at a particular moment in time. The server has a function, `NewGameStateDump`, that iterates over every "live" (real) entity currently being tracked by the server and creates a dump copy of it, collecting them into a `GameStateDump`. All of the data in these structures is immutable, and can be controlled much more tightly than access to the live objects. These game state dumps serve the following purposes:

2.5.1 Game Replays

For each iteration within a round, the server captures a Game State Dump before the first iteration (to record the starting state) and after every iteration. These are exported after each game into a file called `game_dump.json`, which can be loaded at a later time for further analysis, such as is done by the visualizer.

The structure of the game replays is a two-dimensional JSON array. The outer array represents rounds, and each inner array represents iterations within that round. Each inner array element is a Game State Dump, which is encoded as a JSON object. This allows software viewing the data to clearly distinguish how many rounds there were, and how many iterations were in each round.

2.5.2 Statistical Analysis

The server performs some basic statistical analysis using the Game State Dumps captured in a particular game.

For each round, it will calculate the average lifetime, energy, and points of each individual agent, along with the variance of the energy and points. These are written by the server after each game to a spreadsheet named `statistics.xlsx`, which contains a worksheet for each property analyzed. Each worksheet contains a table showing the property of each agent for each round. Each agent is identified by the group number that designed it, with group 0 representing the BaseBiker.

[Image: Spreadsheet]

The server also outputs, at the end of each game, the same statistics averaged over the entire game (every round).

2.5.3 Agent World Visibility Control

The Game State Dumps are used to control what properties of SOMASWorld the agents are allowed to see and interact with. This is done by implementing `IGameState`, which is passed to the agents by the server when their view of the world is to be updated.

Each dump structure partially implements the interface that its live counterpart implements; the `AgentDump` structure, for example, implements `IBaseBiker`. Since the bikers expect to see `IBaseBiker` interfaces when viewing the properties of another biker, they cannot tell and don't care if this is a real agent or an `AgentDump`.

Most of the methods on other bikers, such as `UpdateEnergyLevel` or `GetForces`, should not be callable by other agents. These methods in the `AgentDump` are stub implementations that panic, with a message describing how this function is illegal to call from the agent's context. This allows for tight control over what agents are allowed to do to each other (nothing except via messaging), and what properties about each other they are allowed to view.

Another advantage of this solution is that dumps are immutable; Agents only get an updated world view when the server allows it. This prevents agents from being able to see live data by storing a pointer to a live structure that they were previously given, ensuring that agents are fairly treated by the server. Agents can also store their historical game states to compare them without having to worry about data consistency issues.

2.6 Voting

2.7 Voting Methods

It is important to formulate strategic interaction concerning the aggregation of a set of individual preferences into a collective social preference. Stemming from this premise, we have implemented seven voting methods, based on our lecture notes, to investigate the impact of different voting mechanisms. These voting functions are designed with consistent input and output data structures, facilitating seamless transitions between methods through the modification of a single hyper-parameter. For leadership governed Mega-Bikes, the weighted leadership vote is also included in each method.

- **Plurality:** Each voter selects one candidate. The candidate with the most first-placed votes wins.
- **Runoff:**
  1. 1st round: Each voter selects one candidate. The two candidates with the most first-placed votes are identified. If one already has a majority, that candidate wins.
  2. 2nd round: Each voter selects one candidate. The candidate with the most votes now wins.
- **Borda Count:** Each voter ranks all candidates. With n candidates, a rank k scores $(n - k) + 1$ Borda points. The candidate with the highest Borda score wins.
- **Instant Runoff:** Each voter ranks all candidates. The candidate with the least number of first-place votes is eliminated. This is repeated until only one candidate remains.
- **Approval:** A ballot represents the set of candidates who are 'equally acceptable' to the voter, not a linear rank order of decreasing preference.
- **Copeland Scoring:** Each voter submits a ballot with a linear rank order. A win-loss record, the Copeland Score, is calculated for each candidate.
- **The D'Hondt Method:** The number of winners is proportional to the share of the vote.

Chapter 3: Visualisation

3.1 Motivation

For SOMAS World, a playground consisting of dynamic agents that work together to sustain survival, a visual representation of the world in action is very desirable. Additionally, it is more obvious to see erroneous behaviour than it is to decode erroneous behaviour from a series of console outputs, marking the Visualiser as not just an extension to the SOMAS World, but imperative in its development, testing and debugging. It is for these reasons that we have developed the SOMAS Visualiser, written in Python by Shaheen from Group 5.

3.2 Technical Specifications

The Visualiser had only one goal to fulfil; to display the Agents' interactions intuitively.

3.2.1 Design Philosophy

For the design of the Visualiser, I leveraged my existing understanding of game development, as well as experience gained during ELEC50009 Information Processing, to develop the Visualiser's design philosophy.

The primary game loop is to be short, consisting of mainly function calls to other functions that handle specific tasks. This separation of logic is crucial for debugging, and easily allows for the addition of new features as the Visualiser and platform progressed. The structure of the Visualiser also supported this; the files are named by their purpose and only handle logic that fits that purpose. This approach also greatly benefits during optimisation with a profiler as will be seen later.

I was also sure not to hardcode any constants into the to allow the visualiser to be easily configurable. The following table shows a list of variables that can be configured to customise the Visualiser, contained in "visualiser/utils/Constants.py".

Variable Name	**Description**	**Variable Name**	**Description**
WINDOW_TITLE	The title of the window.	ARROWS	Styling for the directional arrows on entities.
FRAMERATE	The max framerate to run at.	CONSOLE	The colours for the different types of messages on the console.
FPSDISPLAYRATE	The framerate at which to draw the FPS counter and mouse coordinates.	OVERLAY	The styling for the overlay text when clicking on an entity.
MINZOOM, MAXZOOM, ZOOM	The allowable minimum and maximum zoom, as well as the default zoom level.	BIKE	The styling for displaying a MegaBike.
PRECISION	How many decimal places to round numerical values to in the overlays.	Awdi	The styling for display an Awdi.
ENERGYTHRESHOLD	The threshold below which the Visualiser will assume the Agent died from starvation, if the agent has died.	LOOTBOX	The styling for displaying a Lootbox.
THEMEJSON	The path to the theme.json file used to style pygame_gui's widgets.	DIM	The dimensions of the window and the inner game screen.
JSONPATH	The default path of the simulation dump.	BGCOLOURS	The background colours for the main and game screens.
MAXSPEED	The maximum allowed iterations/second for playback.	GOVERNANCE	The different colours for the MegaBikes for a given governance.
		COLOURS	The colour map for the entities.

Table 3.1: Configurable Visualiser Parameters

3.2.2 Technology Stack

The Visualiser is written in Python with Pygame [60] and pygame_gui [35];when planning for the Visualiser before development, I recognised that eventually I would need Pygame's collision detection and event handling system to have the overlay feature when clicking on an entity to display its attributes. Pygame_gui provided the functionality of buttons, sliders, and text boxes, which I otherwise would have had to develop custom classes for. The combination of these two libraries gives a strong foundation to work with and contains (mostly) all the required features needed for the Visualiser.

I chose Python as I have written several tools before in Python and it allows for quick prototyping and adaptable development.

3.2.3 Data Handling

The Visualiser loads the JSON dump from the game upon launch. If unsuccessful, it brings the user to the Main Menu instead and presents a button to load a JSON dump.

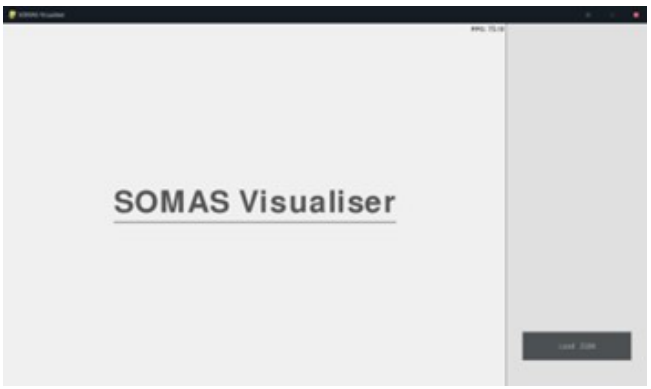


Figure 3.1: Visualiser Main Menu

The JSON is loaded using Python's default JSON library, and the Visualiser simply keeps track of the current round and iteration number to help locate the section of the JSON that represents the current iteration. The JSON dump is made specifically for the Visualiser, so I was able to add attributes as I saw fit. I initially proposed the format of the dump, but it was improved upon by members of the infrastructure team, especially Harry from Group 2.

3.3 Development Process

3.3.1 Challenges and Solutions

The nature of developing an interface using Pygame means that a significant amount of work must be done before any useful information result can be achieved. This meant that the development of the Visualiser must be planned with careful deliberation.

3.3.2 UI Design

At the start of the Visualiser's development, I had a clear goal for what the Visualiser needed to achieve; a button to change the current round, a way to load and process the data dump, an abstract class that provided rendering capabilities to the 4 entities, and a method of displaying entities attributes when clicked by the user. With this in mind, I was able to plan what the UI should look like and had that layout in mind during development; it has stayed relatively constant since; the only changes being additional buttons in the side panel. This has helped to streamline the UI of the Visualiser and prevented unnecessary layout changes.

3.3.3 Grid Drawing and Zooming

On the Visualiser, a grid is drawn that seems to move with the cursor as it is dragged – in reality, the grid is only drawn on the visible canvas. This grid rendering method takes into consideration the mouse coordinates and whether the user is holding the mouse button down to provide a grid that pans naturally. This grid enables a sense of relative position between entities, helping to always gauge the size of the map and the level of zoom without needing to see an entity for visual reference.

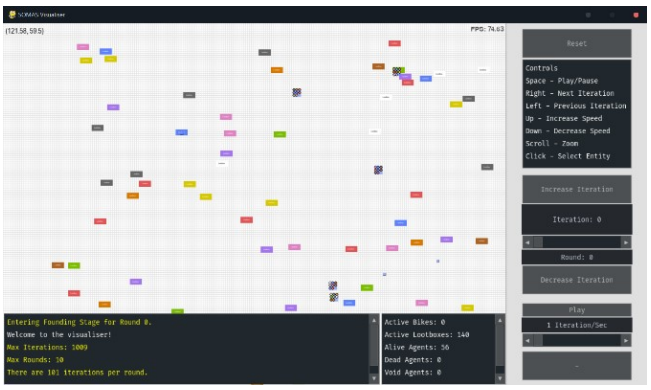


Figure 3.2: Latest Visualiser

Figure 3.3: First Visualiser

Additionally, zooming must be natural; the Visualiser should zoom in on the current mouse position and should accurately scale the grid, adding complexity.

3.3.4 Entity Drawing

All rendering of entities must take into consideration the current mouse position and the amount the grid has been panned to display their relative positions correctly.

The position of an Agent is equal to the position of the bike that it is on, however, these coordinates must be offset when rendering the Mega-Bikes otherwise all the Agents will overlap one another. Due to this, Mega-Bikes must dynamically scale up in size to match the number of Agents in square numbers, ie a Mega-Bike can fit 1, 4, 9, etc Agents, and will expand if the current capacity of the Mega-Bike cannot accommodate the total number of Agents.

3.3.5 Overlay Drawing

For the overlay, all entities inherit the method implemented in the abstract Drawable class they inherit from; there is no need to derive unique methods for each of the entities as the only thing that differs are the displayed attributes.

3.3.6 Arrow Drawing

To display the directional arrows that span the screen, I must know a point on the arrow and either the orientation or second point on the arrow. Once I have these, I can determine the gradient of the arrow, however, I must determine where this arrow intersects with the borders of the game screen otherwise it may draw to infinity, causing rendering to come to a halt. This was done by solving sets of simultaneous equations in the code to produce pairs of x and y coordinates to draw the arrow between.

3.3.7 Version Control

The Visualiser was developed in its own branch in the SOMAS2023 repository, making Pull Requests (PRs) to the main branch whenever new features were introduced. This allowed everyone to easily retrieve a new version

of the platform and the Visualiser that was stable. Given the Visualiser is its own entity, it did not interfere with the functionality of SOMAS World except through the JSON dumper which was specifically designed for the Visualiser, so new updates to the Visualiser did not affect the platform, allowing for new updates to be provided easily to the rest of the agent teams.

3.4 Optimisation

Following the demonstration of the Visualiser on the 8th of December, it was clear to me that the focus of the Visualiser, after completing its main objectives, should be a better user experience in terms of smoothness and stability. At the time, it managed around 10 FPS during playback, resulting in choppy playback and an increased likelihood of crashing if several complex events happened in advance. This directly affects the user experience but also poses challenges in accurately tracking and analysing agent behaviours in real-time.

Figure 3.4: SnakeViz's processing of data produced by cProfiler.

Given the Visualiser's design philosophy of delegating code execution to separate functions, a profiler would be able to identify which function calls were the most expensive, allowing me to make optimisations that shorten the game loop, thereby increasing the frame rate of the Visualiser. For this, cProfiler and snakeviz [69] (to analyse the profiler's output) were used to profile the Visualiser's function calls and were incredibly efficient in showing where optimisations needed to be made.

Figure 3.5: Analysis of run_loop()

Delving deeper into the tree, the game loop is made up of two major sections: the "render_game_screen" function and "handle_events" function. Immediately it becomes obvious that "ui_text_box"'s rebuild method (in highlighted purple) was consuming most of the time for the "handle events" function, responsible for displaying the console. It was clear that this method is being called far too many times during the game loop, and I should call this method much more sparingly. SnakeViz's processing of data produced by cProfiler.

Analysing the "render_game_screen" calls, it also becomes incredibly clear that most of the call is spent on Pygame's font rendering method, which simply draws text to the screen. In this version of the Visualiser, text is newly rendered every frame, used for the entity labels (the Awdi, the Agents and the Lootboxes), as well as the attribute data displayed when clicking on them, the FPS counter, as well as the mouse coordinates. The optimisation was made so that text is only redrawn if their apparent size changes, or whenever the user zooms the screen. This version of the visualiser ran at around 10 FPS and could not surpass speeds of 12 iterations/second for the auto-play function, otherwise, there was a serious risk of crashing.

Post optimisation, the contribution of the "render_game_screen" function to the game loop time has been decreased, however, the bulk of the "handle_events" function is still "Pygame_gui"'s rebuild function. This is

Figure 3.6: Analysis of render_game_screen()

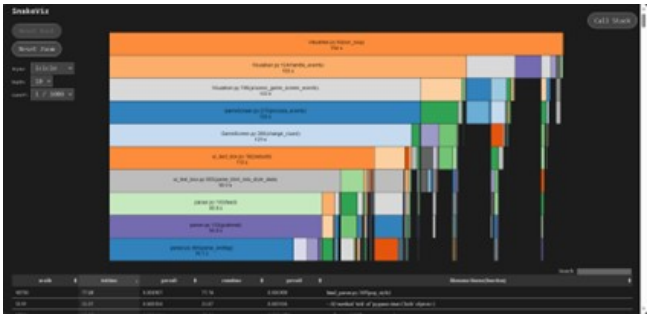


Figure 3.7: Post Optimisation

seen to be called in "change_round" (later called "change_iteration" as naming changed), and this is because the console and statistics must be redrawn every iteration to update them, showing information about the current iteration. An optimisation could be made to combine the console and statistics elements, but this may result in a less aesthetically pleasing interface and the work required may not be worth the performance improvements.

In the new optimised version of the Visualiser, I can uncap my framerate and operate at 100fps at 12 iterations per second, a performance improvement of 10x. This Visualiser can operate at more than 50 iterations per second on my machine.

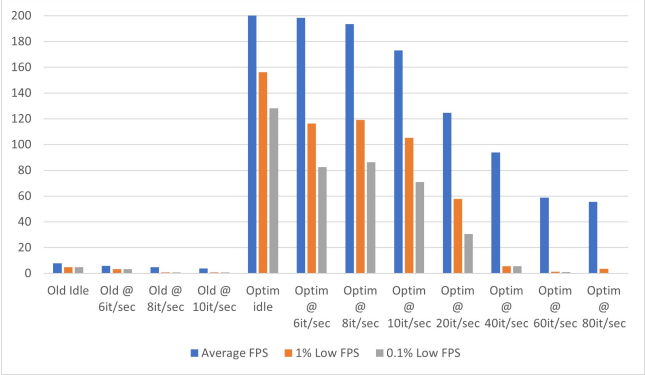


Figure 3.8: Visualiser Performance Metrics

3.5 Impact on Development

The Visualiser's impact on the development of SOMAS World has been significant; there have been several bugs that have been identified using it and either reported to the infrastructure team or were a result of a flaw in an Agent strategy, allowing that team to quickly debug their Agents and fix the erroneous behaviour.

An example of a bug that was made clear to the Infrastructure team was that the initial calculation of the turning force of a Mega-Bike was determined from the average of all the contributions of the Agents on the Mega-Bike. This value was from -1 to 1, mapped to -pi to pi, or from -180 degrees to 180 degrees. This means that both -1 and 1 correspond to the same direction. Consider the scenario where two Agents share a single Mega-Bike, and both agents wish to turn backwards, one advocating for a -1 turning force and the other a 1.

They both intend to turn around but the turning force that is calculated for the bike is 0, forcing the bike forwards, easily leading both these Agents to starvation as they attempt to change the Mega-Bike's course.

This would be incredibly difficult to debug without the Visualiser, as it requires knowing the intentions of the Agents on the Mega-Bike, the overall steering of that Mega-Bike, as well as how the position of the Mega-Bike changes relative to other entities in the map; this was incredibly clear as the Mega-Bike was running away from the Lootboxes on the map, which is guaranteed death for the Agents. Correcting this behaviour would have been difficult if it was not being deliberately looked for if the Visualiser was not present.

Figure 3.9: Mega-Bike unwillingly running away from the map.

In a similar vein, the Visualiser has allowed Agent strategies to develop as expected. It is easy to see the behaviour of a Mega-Bike consisting of Agents from a particular team at a macro level. The ability to display each Agent's attributes also helps to confirm that the Agent behaves as expected.

3.6 Results

As mentioned before, the Visualiser can perform at high frame rates and heavy load (with a high iteration/second during playback), fulfilling its objective as a tool that informs about SOMAS World whilst maintaining a good user experience.

When replaying SOMAS World through the Visualiser, it is clear that the Agents from different teams, with different strategies, are able to work as a cohesive system, cooperating on the Mega-Bikes to sustain survival, affirming that the platform delivers on its purpose.

3.7 Future Work

The Visualiser, although a strong tool, could benefit from improvements. In terms of optimisation, a new method for the text drawing would be highly beneficial, as this seems to be the largest bottleneck in its gameloop.

While the Visualiser currently has support for directional arrows for the Awdi and Mega-Bike, it currently does not support it for the Agents; the code exists for the agent but is flawed and comes from a misunderstanding of how the Agent's steering forces contribute to the total steering force of the Mega-Bike in that round.

Additionally, another useful feature would be to track a given entity with some hotkey, preferably on the mouse. This would likely be quite trivial to implement given how the entities are drawn and all are subclasses of a baseclass called 'Drawable' – the logic would be implemented here and would simply change the screen's location according to the current position of the tracked entity.

A final improvement would be to create a data handling class; usually, the Visualiser only supports the most recent version of SOMAS World and does not usually have backward compatibility with older JSON dumps. A data handler could bridge this gap and would allow the Visualiser to display data shown in older JSON dumps and would simply not the attributes that are missing, instead of refusing to load entirely. This would require significant rework of the Visualiser.

3.8 Conclusion

I am incredibly grateful to the rest of the Infrastructure team; their diligent work created a stable foundation from which I was able to focus on the development of the Visualiser and deliver a strong tool. I am also grateful

to the capable members of Group 5, for developing the code for my team's Agent strategy. Without their contributions, it would have been difficult to develop the Visualiser without worrying of the platform's, and my own agent's, functionality.

Chapter 4: Project Management and Devops

4.1 Project Organization

As part of a Multi-Agent System ourselves, we needed to establish some formal organization of the class and the project and communication channels all of which were of utmost importance for the progress of the project.

4.1.1 Communication

Our main source of organization was through in-person and online meetings, both General and Team-Based ones. Apart from having weekly meetings, we decided to use the communication platform "Slack" as the main channel for global and intra-team communications. Each sub-team had a separate channel to communicate, discuss and share ideas that could be brought up for discussion on the general meetings we had so that a final decision was made. As far as dispute resolution is concerned, in most cases we went with majority voting upon the ideas that were up for discussion. In some other occasions (such as General Rules and Conventions) we went with general consensus: ideas where brought up and if no one objected, those were added to the list of rules/conventions we needed to follow.

4.1.2 Documentation

Each sub-team was responsible for keeping up-to-date documentation of what has been implemented/agreed upon as a reference for everyone on the project (e.g., Overview Documentation, Maths Documentation, Infrastructure MVP and Quick Start Documentation). Meeting notes were also made during discussions and progress check-in sessions every Friday.

4.1.3 Rules and Conventions

In order for us to effectively work in parallel towards accomplishing our common goal (i.e., develop a platform where agent strategies could be formed to collect Lootboxes) we had to agree and decide upon some General Rules and Conventions. Those included General Project Management Rules, Setup and Rules for Implementation, all of which are publicly available on the main page of our projects' repository and in docs folder in our repository.

4.1.4 Programming Language

Upon voting, the majority agreed to use the programming language of Go. That decision was heavily influenced by Go's ability to implement concurrency which was a desired feature for the Peer-2-Peer communication level between the agents. The availability of the SOMAS Base platform built using Go was another major factor. Some other important factors were previous years' implementations and package dependency management (with go.mod).

4.1.5 Project Management

One of the most crucial parts in any Software Engineering Project is effectively tracking changes on files. As a solution to that we have utilised Git as a Version Control System and GitHub as the online counterpart.

Github

The project code repository can be found on: https://github.com/SOMAS2023/SOMAS2023 This repository will be hosting the code of our Environment and Agents' implementations. On this repository, each team (either sub-teams or agent teams) have created a separate branch for a specific feature and worked on it. After the implementation was done they have created a Pull Request to merge the changes to the main branch. We have agreed to use branch protection on the main branch and each merge was only possible after it has been reviewed by at least another member that was not involved in the implementation of that specific feature, to ensure code integrity and readability. This was also meant to reduce the number of bugs in the code overall.

Project Progress

In order to track Project Progress and Task Allocation we have used GitHubs' Issues feature. Through that functionality we were able to Create and List Tasks that needed to be done by hierarchically ordering them according to importance and team-specific labels. Moreover, members of the Infrastructure Team could self-assign to a single (or multiple) issue which helped improve the efficiency and transparency for each task. Milestones and deadlines were agrred upon in meetings and communicated through the respective slack channels and during in person meetings.

4.2 Devops

The foundational idea behind the DevOps Team was to make the implementation process easily accessible to everyone while ensuring that the quality and integrity of code, according to communally agreed upon "best practices", created at any step was not violated. These considerations stemmed from the fact that many people in teams had little programming experience and/or little to no Golang experience.It also stemmed from the fact that multiple people will be working on different features and there needs to an organized approach to ensure no task was repeated or forgotten.This was accomplished through various Continuous Integration- Continuous Deployment (CI-CD) methods:

4.2.1 Github Actions

GitHub Actions are a GitHub feature that allows for the automation of repetitive tasks, such as building, testing, and deploying code, thus making it a powerful tool for automating development workflows. We used this as not only does it save time and effort for teams, but it also helps to ensure that the software is always in a deployable state. With automated builds and tests, the agent teams can focus on developing and testing their agents, rather than worrying about manually running builds and tests every time they make a change, or having to use Linux scripting tools they might not be used to. GitHub Actions allows for the creation of custom workflows that are triggered by specific events, in our case the creation of a new branch or the opening of a pull request. This allowed us to automate the review process, ensuring that code is tested before everyone reviews it and it gets merged to the main branch.

As part of our workflow, our code was complied and our tests were automatically run, but also any changes went through linting, deadlock detection, and cyclic reference checking: • Linting helped us find syntax errors, formatting issues, and potential bugs, which make code easy to read, while increasing its quality and maintainability. • Deadlock detection allowed us to identify and prevent situations where two or more channels are waiting for resources that are held by the other channels, which is critical to our system due to our heavy use of concurrency. This increased the performance by avoiding indefinite loops and the reliability by avoiding our program timing out. • Cyclic reference checking let us detect and resolve circular references in data structures, which directly improved memory usage, reliability and maintainability. From a self-organizing perspective, another benefit of using GitHub Actions is that it provides a centralized, non-biased and consistent way to manage the software development process across the infrastructure team. With comments about errors and mistakes now automated, interpersonal conflicts arising due to arguments are now automated away.

4.2.2 Unit Testing

Due to the heterogeneous nature of our project, and due to the different implementations of each agent team, our testing strategy consisted of unit testing, i.e. testing individual units of code at the function or module level for proper behavior. This was to ensure that despite any code change, our core infrastructure functionality would function. This allowed teams to delve deeper into their agent testing and self-organizing ideas as our core code base was now more maintainable and robust.

For example, each new feature had unit tests written to handle edge cases. This ensures the repeatability and reliability of our infrastructure platform.

4.2.3 Scripts/Deployment

Just use the Terminal to run Go with the following approaches:

- **Approach 1** - For Linux and macOS users:
  - Run the command: `go run .`
  - Use `sudo go run .` if you encounter any "Permission denied" errors.

- **Approach 2** - For Windows users:
  - First, perform a build step with the command: `go build`
  - Then, execute the binary: `./SOMAS2023` (or `SOMAS2023.exe` on Windows)
  - Use `sudo` on Linux and macOS, the same as Approach 1 if required.

4.2.4 Package Management

Our programming language Go was also picked for its DevOps advantages, more specifically when it comes to package management, which is very prone to many issues such as bugs and unreliability. This combined with Go's fast compile times creates a fantastic development environment for everyone to use.

Firstly, Go has a built-in dependency management system that allows us to specify the dependencies of their packages in a standardized way, which makes it easy to manage and update dependencies as needed, and most importantly, consistent across all members.

Go's dependency management system also includes support for versioning, which allows us to specify the specific version of a dependency that a package requires, again ensuring reliability of the project.

4.2.5 Project Structure

Following common coding practises in Go and project structures from previous years as our main inspiration, we came up with the following structure in 4.1 :

```
/
├── docs/
│   └── documentation files/
├── internal/
│   ├── clients/
│   │   └── Individual Teams agents/experimentations/
│   ├── common/
│   │   └── Common utilities, or system-wide code/
│   └── server/
│       └── Infrastructure implementation and unit tests for each feature/
├── visualizer/
│   └── Visualization implementation/
├── go.mod and go.sum/
│   └── Calculate and Output statistics per round in Excel file/
├── main.go/
│   └── Executes the game/
└── README.md/
    └── Provides details about the SOMAS2023 project/
```

Figure 4.1: SOMAS2023 Project Structure

Chapter 5: Team 1

5.1 Strategy

Our agent strategy is centered around relationships, fairness and dynamic decision-making given changes in the environment, bike compositions and social objectives. The heart of our agent's approach lies in a Beliefs, Desires, Intentions (BDI) framework and the intentional stance proposed by Dennett [16], providing our agent with predictive leverage by interpreting the intentions and behaviours of other agents within SOMAS World.

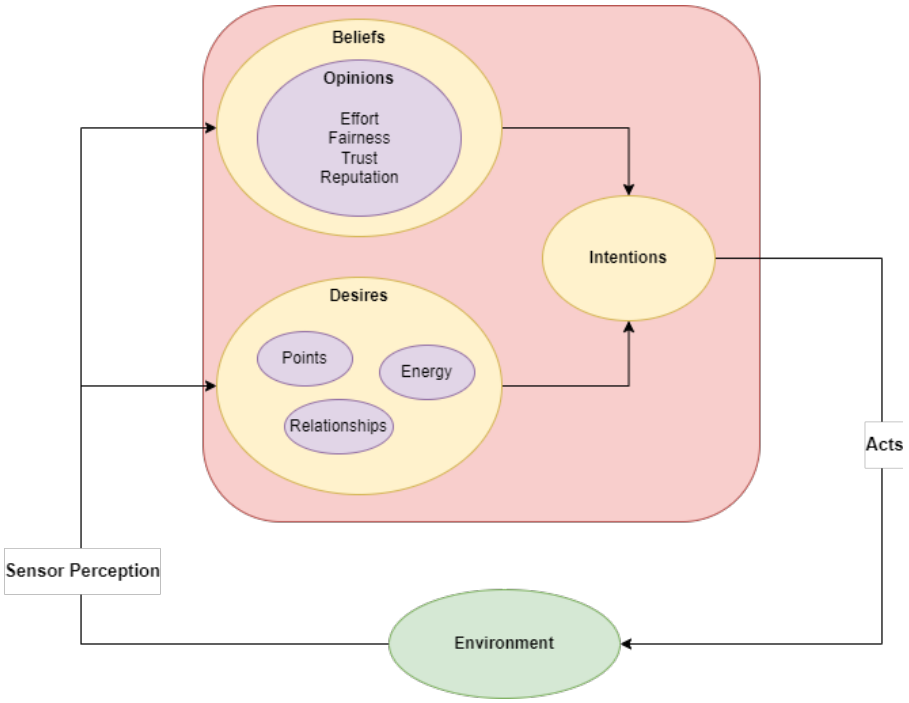


Figure 5.1: Agent 1 BDI Framework

Taking inspiration from Maslow's hierarchy of needs [38], we designed our agents decision-making process around the fulfillment of self-actualization needs, namely energy, points and relationships with other agents. In conjunction with the BDI framework, our agent operates using feedback loops for evaluation and adaptation to shifts in other agent behaviours and the environment. Our strategy emphasises the assessment of fairness, trust and cooperation, rewarding those that favour equitable fulfillment and punishing those that deviate for their own self-interests. Aligning with the canons of distributive justice [65], we ensure the acknowledgement of contribution and cooperative interaction without disregarding basic needs, in effort to encourage cooperative behaviour that fairly benefits all.

5.2 Implementation

5.2.1 Opinion Formation

A key component to decision making in this game relied on a dynamic and complex opinion structure. Opinion formulation for our agent is structured as a weighted function comprising three integral components: effort, fairness, and trust. This opinion framework holds immense significance in guiding our agent's actions, due to our agents' focus on cultivating relationships and its view of the game as a collective action problem.

In the game, our agent believes that there is a collective responsibility for agents to prioritize the survival of all individuals. It upholds a belief in adhering to collectively agreed-upon decisions made by the bike after a vote. This societal perspective shapes our agent's opinion construction, contributing to the establishment of societal ideals. Negative consequences (a deduction in opinion) are imposed on individuals deemed unfair,

untrustworthy, or lacking in productive contribution. Simultaneously, our agent forges positive relationships with like-minded agents who share similar societal values.

For instance, an agent consistently engaging in free-riding on the same bike as ours triggers a decline in our agent's opinion. This diminishing opinion may lead to our agent proposing the removal of the free-rider from the bike, or our agent may be less likely to support their desires or needs regarding Lootbox nominations and distributions. Furthermore, in scenarios where the average opinion of the entire bike becomes unfavourably low, our agent chooses to exit, seeking a community with a more reasonable average opinion, as discussed in Section 5.2.5.

The foundation of this opinion construction draws inspiration from Berger and Luckmann [6] aligning with their exploration of how social structures and institutions shape individuals' perceptions and actions within a shared reality. The interplay of effort, fairness, and trust encapsulate our agent's commitment to fostering a harmonious virtual society grounded in shared values and cooperative principles.

Effort

The inclusion of effort in shaping our agents' opinion formulation is driven by our agent's fundamental belief that all participants on the bike are expected to contribute to its movement. Considering effort within the opinion framework introduces consequences for those who engage in free-riding, guiding our agent to gravitate towards bikes where reasonable effort is collectively made by each participant. This strategic orientation benefits not only our agent but also the entire bike community, facilitating quicker access to Lootboxes with a manageable energy strain on all individuals. Effort is calculated by a prediction of the pedal force of each agent. While the actual pedal force of each agent isn't within the knowledge sector of our agent, the total force used by the bike can be estimated by our agent, which enables predictions to be made. These predictions, based on colour alignment, energy levels, and current trust ratings, stem from the ingrained beliefs that drive our agent's decision-making process.

If an agent's colour does not align with the colour of the Lootbox the bike is heading towards, they are more likely to use less pedal force than an agent whose colour does align with the targeted Lootbox.

An agent with a lower energy level is likely to use less pedal force than an agent with a higher energy level.

A trustworthy agent is less likely to act against the agreed upon decisions of the group (for example by free-riding), than a non-trustworthy agent.

Using these three beliefs, the agent uses the following calculation.

If the agent's colour is not the colour of the Lootbox that the agent is heading towards, the colour probability increases to 50%. This means that our agent believes that there is a 50% likelihood for an agent with a colour that does not align with the Lootbox colour to pedal less than the desired pedal force.

The agent's energy need is also considered in the function: an agent with an energy of 0.2 has an energy need of 0.8, that means that our agent believes that agent is 80% likely to not pedal at the desired, mutually beneficial pedalling force.

These two values are combined to produce an 'effort probability', which is then scaled proportionately to the remaining total pedal force after removing our agent's own contribution. This process provides an estimate of each agent's contribution to the remaining pedal force, approximating a prediction of their individual pedal force.

The 'desired, mutually beneficial pedalling force' that our agent expects of other agents is currently set to our agent's own pedal force: our agent expects that other agents should at least be pedalling what it is pedalling itself.

The calculated increment or decrement in effort is determined by calculating the difference between the predicted pedal force with the expected desired pedalling force. Using the calculated effort probability, we appropriately scale this difference in pedal force, to ensure that agents are not penalised for their circumstances such as low energy levels. Importantly, existing trust scores are factored into these adjustments, since the predictions are not absolute truths. Trust weights play a crucial role in rewarding trustworthy agents, ensuring that they are less likely to be subject to doubt. This inclusion of trust prevents overly penalising agents who have previously demonstrated trustworthiness in interactions.

The consideration of effort aligns with the 'actual productive contribution' canon of distributive justice, providing a legitimate claim for our agent's opinion formulation.

$$\textit{Effort Update} = (c + e_n) * \frac{t_{ee}}{t_{ep}} * E_{diff} * T \tag{5.1}$$

where:

c = Likelihood of an agent not pedalling the expected pedal force based on the colour of the Lootbox
e_n = Energy need of agent
e_p = Effort probability ($e_n + c$)
t_{ep} = Total effort probability: sum of all calculated effort probabilities
t_{ee} = Total expended energy used by the bike in that iteration
T = Current trust score of agent
E_{diff} = Difference between biker's expended energy and our agent's expended energy

Fairness

In our agent's societal view, a fair agent is one committed to the equitable distribution of loot, meticulously considering the energy needs of all individuals on the bike. This addresses the Tragedy of the Commons dilemma [21] which arises when individuals acting in their self-interest deplete shared resources, leading to a collective loss by penalising the over-exploitation of shared resources through agents' selfish actions. The fairness of an agent is calculated by comparing each agent's energy gain from the Lootbox to a sensible approximation of the Pareto Optimal Lootbox allocation outcome. The determination of the 'fairest' Lootbox distribution involves a calculation of each agent's energy need in order to reach maximum energy, normalized to a proportion of the total Lootbox energy. This ensures that the distribution is equitable, with agents' struggling with low energy ideally gaining more from the Lootbox than those with higher energy levels. This was influenced by the 'Difference Principle' of John Rawls' Theory of Justice [11] which proposes that individuals designing the rules for a society in which they are unaware of what their own circumstances will design a just society, maximising the well-being of the least advantaged individuals.

The knowledge sector of our agent includes the energy levels of other agents within the game, providing a comprehensive understanding of other agents' energy needs and gains and losses. Once a Lootbox has been obtained, our agent calculates the disparity between each of the other agents' energy gains and the 'fairest' Lootbox allocation outcome. This disparity, whether positive or negative, is then incorporated into the agent's current fairness score. This incorporates the 'efforts and sacrifices' canon of distributive justice in our agent's opinion formulation: agents that are more generous, distributing less energy than they need to themselves in order to give others more (sacrificing their own energy), are rewarded, whereas those that take instead of give are punished.

The fairness component further influences our agent's strategy to vote leaders into governance. In a 'democracy' governance type, where each agent's vote equally influences Lootbox distribution, accountability becomes paramount. Changes in fairness scores hinge solely on the disparity in an agents' own allocation. On the other hand, within 'leadership' and 'dictatorship' governance types, only the leader's fairness is impacted by distribution differences, reinforcing accountability within leadership roles. Unfair leadership or dictator behaviour incurs consequences in our agent's opinion of them, diminishing our agent's likelihood to vote for them in subsequent rounds, and aligning with the transparency principle.

Trust

The trust component in our opinion formulation draws upon the established trust framework [55].

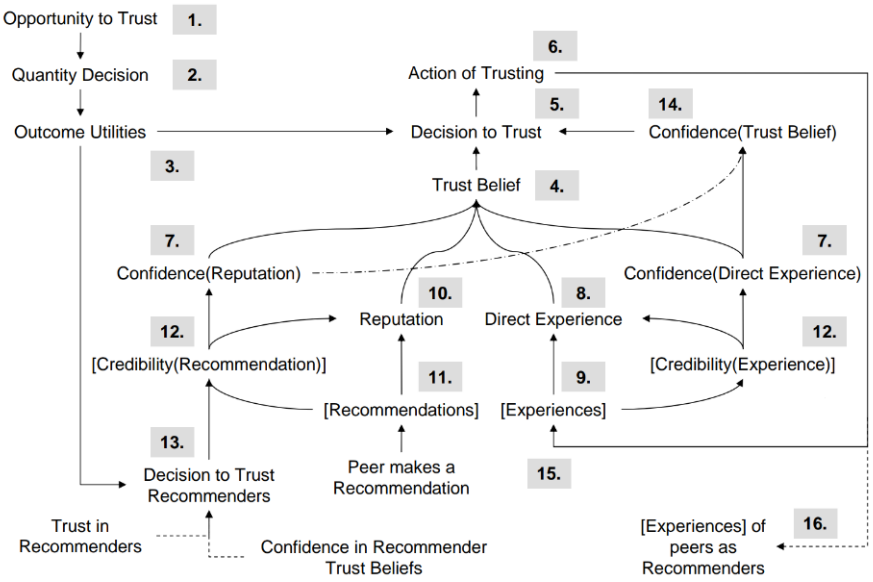


Figure 5.2: Trust Framework

This component is split into two parts: direct experience and reputation, with the combination updating in a feedback loop. The direct experience aspect hinges on the deviation of the bike's steering angle from the intended direction, representing the normative dimension of trust. Our agent, adhering to the belief that the bike's direction is socially constructed and should be respected, deems agents deviating from this norm as less trustworthy. While the forces of other agents are not part of our agent's knowledge sector, the current orientation of the bike is. Calculating the new orientation in relation to the intended direction, our agent predicts potential deviations based on colour alignment with the target Lootbox and the current trust in the agent. Current trust serves as a confidence contributor to direct experience – trustworthy agents are less likely to deviate, and a weight is incorporated to the final deviation prediction.

Reputation, constituting the cognitive dimension of trust, is perceived by our agent as the global opinion of an agent within the virtual society. Agents with favourable opinions are deemed to have a good reputation, while those with unfavourable opinions have a poor reputation. Reputation assessment is facilitated through the messaging component (see Section 5.2.2). Trustworthy agents (those with a trust value that is greater than a minimum trust constant threshold) that broadcast opinions, increment or decrement an agent's reputation. The credibility of recommendations is carefully considered, with only opinions from trustworthy agents contributing to reputation changes.

This trust framework ensures that our agent leans towards 'reliance trust' instead of 'risk trust' [55]. This preference is based on the significant importance assigned to observed behaviour and the general societal opinion, both of which play a pivotal role in shaping the agent's decisions related to trust. Operating within a BDI (Belief-Desire-Intention) framework, our agent's decisions heavily depend on its current beliefs. Therefore, it becomes essential to safeguard these decisions from the influence of untrustworthy or narcissistic agents who might act solely in their own self-interest. Our agent intentionally avoids allowing such agents to heavily impact its decisions, aiming to uphold a robust and trustworthy decision-making process.

5.2.2 Messaging

Verifying Senders

Our agent's messaging system is set up to only accept messages from agents deemed trustworthy. Given the presence of narcissistic agents and agents that lie for self-serving reasons within the game, our agent adopts a precautionary approach. It first ensures that the sending agent possesses a trust score surpassing a predefined threshold constant of 0.7. Additionally, our agent requires that its opinion of the sender is greater than 0.5. This verification process is implemented to mitigate against detrimental effects of false information on our agent's cognitive framework, influencing its beliefs, desires and subsequent actions.

Removal of Agents

Upon receiving a message from a trustworthy agent proposing to kick another agent off the bike, our agent considers its opinion of the agent that has been proposed for removal. If the current opinion is positive (above 0.5), the sender's opinion will be slightly penalised by a 10 per cent decrease in our agent's opinion of them. This penalty is because their desire to remove an agent from the bike who appears to be overall positive is a negative mindset, contrary to our agent's ideals of a cohesive, collaborative society. Conversely, if the opinion of the targeted agent is lower than 0.5, the opinion of the sender will be improved by a 10 per cent increase, since there is now further support that the sender and our agent share a similar mindset and similar values. The adjustment of the opinion of the sender based on our agent's opinion of the proposed target emulates elements of social identity theory [71]. This theory suggests that individuals categorize themselves and others into social groups based on shared characteristics and attitudes.

Reputation

When an agent receives a Reputation message from a trustworthy peer, it prompts a twofold update. Primarily, the trust value associated with the agent mentioned in the message is updated by the reputation value sent, multiplied by a reputation scaling constant, to align with the established trust framework discussed in Section 5.2.1 This scaling constant represents how easily influenced our agent is in the context of its opinions of other agents. The reputation value of the specified agent is also updated

Requests to Join

When our agent receives a message from an agent that requests to join the bike, our agent first checks if the trustworthiness of that agent and our opinion of them is above a specified threshold (0.5). This is different from the Verify Senders method since that method only considers senders that are already on the same bike as our agent. If that agent shares the same colour as our agent, our agent's opinion of the sender is improved by 10 per cent. This ensures that our agent is more likely to vote them in during the voting stage of accepting agents onto the bike (see Section 5.2.5). When more agents on the bike share our color, there is a higher likelihood that a Lootbox of our color will be the agreed upon target Lootbox. This further increases the likelihood of high productive contribution from agents, in anticipation of receiving rewards (points from their colour Lootbox),

Lootbox Choice

The agent responds to messages from trustworthy senders about the Lootbox they would like to go to, by checking if the colour of the Lootbox is the same as the agent's own colour. If so, the agent's opinion of the sender will improve by 20 per cent. This strategic alignment based on matching colours can be seen as a cooperative move within a larger strategic framework. Aligning with agents with the same goals such as choosing the same colour Lootbox, leads to mutual benefits. In this context, aligning with agents who share the same colour preference enhances the probability of our colour Lootbox securing majority votes during decision-making processes.

Handle Forces Message

In a leadership or dictatorship governance, if our agent is the biker, it will respond to other agent's sending their forces as a message. if an agent admits to braking (actively preventing the bike from moving) or turning the steering wheel (causing the bike to deviate), they are immediately added to the kick list to penalize deviations and maintain the bike on the agreed-upon path, safeguarding the collective interests of the bikers.

Conversely, if an agent is found to be not pedalling at all, our agent adjusts its opinion of them downward. While they may not be added to the kick list, this acknowledges the possibility that some bikers may choose not to pedal temporarily to conserve energy. This reflects an awareness of individual contributions to the collective effort, balancing the need for adherence to group goals with a consideration for potential variations in individual energy preservation strategies.

Sending Messages

In the context of sending messages, our agent creates a list of trusted recipients (agents with a trust score greater than the trust threshold 0.7). Our agent only sends messages to agents in this 'Trusted Recipients' list. Due to this, our agent is always truthful in its messages since it has reduced the risk of sending information to untrustworthy or narcissistic agents who could use this information to harm our agent. Our agent sends messages to trustworthy agents regarding its preferred governance and its desired Lootbox, as well as when it is requesting to join a new bike and desires to remove an agent from the current bike. The preferred governance

message is sent to other agents in the founding stages of each round, to increase the likelihood of residing on a bike with agents that desire the same governance type as our agent. The desired Lootbox message is sent to trustworthy agents on our bike.

Create Kick Off Message

Our agent sends a message to trusted recipients saying how they would like to kick somebody off the bike if our agent's opinion of that agent goes below the predetermined kick threshold (which is set to 0.1). An opinion falling to such a low threshold signals that the targeted agent has been engaging in behaviours deemed detrimental to the collective well-being of the bike. Our opinion formulation structure penalizes untrustworthy, unjust, and uncooperative actions, creating a dynamic where actions perceived as putting the bike at risk lead to a potential expulsion through this communicated threat.

5.2.3 Governance

During the governance nomination process, our agent determines its desired governance by examining its current relationships, as well as predicting its own reputation (the method of which this is predicted is discussed below, in Section 5.2.3) and what other agents could think of it at this point in the game.

Our agent considers two extremes when evaluating relationships to determine its' governance nomination: very poor or mediocre relationships typically lead to our agent favouring a democratic governance model. If our agent is generally disliked, the likelihood of being elected into a leadership position is diminished. In such cases, there is a potential risk that the chosen leader might act against our agent's interests, for instance, by allocating less loot than necessary. When our agent holds poor opinions of other agents, it implies a perception that these agents may be untrustworthy, lazy, or unfair due to our opinion formulation structure. Consequently, our agent deems a democratic governance system more favourable than a leadership or dictatorship structure, in this situation.

The alternative scenario our agent considers involves highly successful relationships. When our agent is well-liked or holds strong positive opinions about other agents on the bike, a preference for a dictatorship governance model is likely. Being well liked increases the likelihood of our agent being voted in the leadership role. Moreover, even if our agent is not elected, strong positive opinions of other agents on the bike suggest that the other agents are quite fair, trustworthy, and productive so the elected leader is likely to have these values.

The government nomination method uses thresholds to determine if the average relationships and our agent's predicted reputation is high enough to nominate dictatorship or low enough that a democracy nomination is required. In cases where relationships and our agent's reputation fall within the intermediate range defined by these thresholds, a leadership government type is nominated. Leadership government types balance majority preference and relevant expertise aggregation, reducing the likelihood of majority tyranny that could occur in a democracy government when the majority of agents are one colour.

Our opinion framework penalizes leaders deviating from the shared values of the group during their tenure. This ensures that our agent is less inclined to vote for such leaders in subsequent rounds. This dynamic underscores the significance of alignment between leadership actions and the collective values of the group in our agent's decision-making process when nominating a leader (see Section 5.2.1). Since the leader is held accountable through the opinion framework, in our agent's perspective it is risky for the leader to only prioritise themselves and agents sharing their colour.

Our Reputation

The reputation of our agent is predicted by examining the relative success of our agent in comparison to other agents in the game. If our agent is very successful at the current stage of the game, with the highest number of points compared to all other agents, and high energy levels, whilst another agent has extremely low points and low energy levels, the relative success function would predict that the second agent would not have a high opinion of our agent. From the perspective of our agent, the game is a collective action predicament, where all agents must obtain shares of a common pool resource (specifically Lootboxes, which contain both energy and points). The macro-level goal of the game therefore, in our agent's view, is sustainability. This means that the goal is for most agents need to sustain adequate energy levels to ensure continuous functionality. In the event that one agent obtains extreme levels of success in comparison to the other agents, our agent anticipates a negative perception from others, and a poor reputation. This anticipation is based on the belief that such an extraordinary lead suggests that an unjust act must have occurred for a single agent to significantly outpace others in a game where the emphasis should be on prioritizing agents with the greatest needs.

Our agent uses the relative success method to predict its reputation amongst other agents.

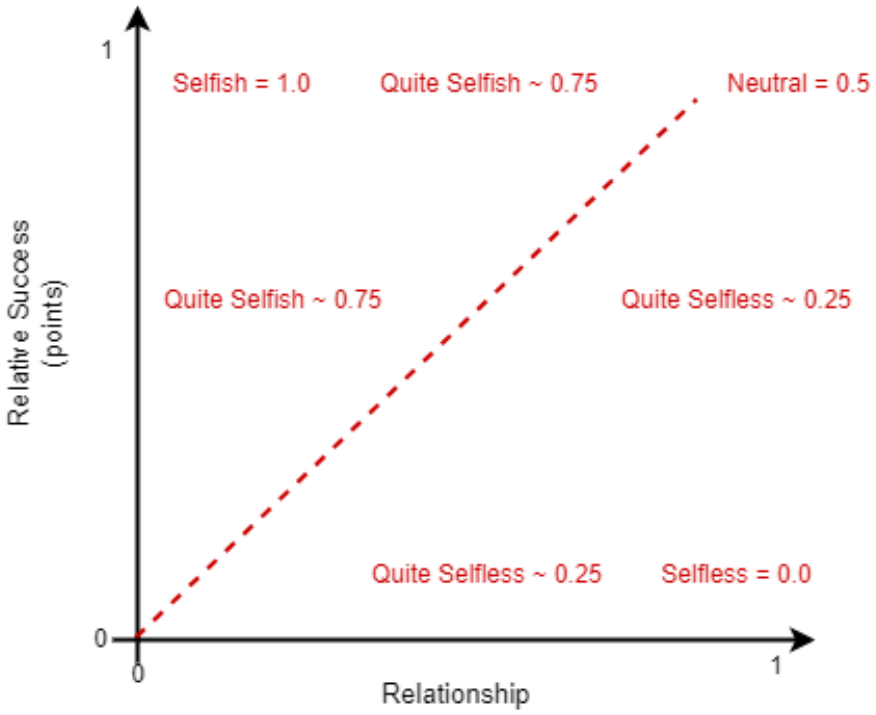


Figure 5.3: Selfishness interpolation

5.2.4 Loot Distribution

Our resource distribution method is built upon our BDI framework. Fundamental facts, such as the success of other agents or deviations in pedalling behaviour shape our overall opinion of other agents, ultimately affecting our desires and voting behaviour for resource distribution. We designed this by utilising the canons of distributive justice, in order to abide by the psychology of our agent as one whose core aim is to achieve fairness. The legitimate claims were to consider agents as equals, according to their needs, their actual productive contribution, efforts and sacrifices, and ability, merit and achievements. The following outlines the steps taken by our agent to determine this distribution.

1. Calculate a selfish and helpful distribution for every agent, including self. These act as prior bounds of distribution without assessing our specific beliefs.
    - Selfish distributions: Prioritise own need first, allocating resources to satisfy entire self need. Distribute remaining resources to others proportional to their needs.
    - Helpful distributions: Distribute resources proportional to everyone's needs, in alignment with 'as equals' and 'needs' canons of distributive justice.
2. Obtain a selfishness score for every agent, including self. Excluding ourselves, this score is an equation of relative success, or point difference, and our opinion/relationship of the agent (see Section 5.2.1) which encapsulates our trust, fairness and effort metric. This highlights the efforts and achievements of the particular agent. Selfishness scores are obtained by the position on the graph below given these 2 values. Intuitively, sitting on the diagonal results in a neutral score, with an indifference in acting selflessly or selfishly. If an agent is dominating in points and we have a poor relationship with them, we have stronger intent to act selfishly, with no motivation for them to obtain a large allocation of resources, hence obtaining a selfishness score closer to 1.0. The same logic can be applied for other regions of the graph. The selfishness score for ourselves is inherently the average of the selfishness scores we calculated for the other agents. Consider a scenario where there are 6 other agents. If we act selfishly towards 3 of them and selflessly towards the other 3, the average of our behaviour towards them results in a neutral outcome.
3. Interpolate between the lower and upper distribution bounds for each agent, given the corresponding selfishness score. If our agent has selfish intentions to an agent, it will vote for a distribution closer to the 'selfish' bound of that agent.
4. With the interpolated distribution values of each agent, we normalise them between 0 and 1 as to obtain a final distribution that sums to 1.

Overall, this approach is designed to consider the relative efforts and contributions of others, which are implicitly captured by the relationship/opinion structure, without the disregard of the fundamental needs of others. Relative success and relationships provide objective and subjective components to our agents' voting. With the BDI feedback loop, relationships are updated frequently for responsive voting decision making.

5.2.5 Bike Membership

The main components of bike membership are those affecting other agents and those that affect ourselves, the former constituting of acceptance of new bikers and kicking of existing bikers, and the latter our decision process behind leaving and joining bikes.

Acceptance

Our agent's decision behind accepting others on our bike arises from two independent thresholds. The first is a holistic decision, where we accept bikers that have the same colour preference as us. While our opinion of them may initially be less than neutral, and likewise us with their opinion structure, most likely the shared desire to obtain Lootboxes of the same colour will result in co-operative and equitable intentions, progressively improving each others opinions in the long run as well as attracting others of the same colour to the bike. This aligns with our desire to maximize the number of points we have while upholding strong relationships.

The agent's second deciding factor on accepting a biker is closer oriented to our opinion structure. We observe our average opinion with all our fellow bikers, leveraging our social network to rationalize how the incoming biker may impact our performance. As mentioned in Section 5.2.1 and the importance of trustworthiness for collective action in [51], we concluded if our average opinion of the current bikers is greater than our opinion of the incoming biker beyond a threshold, then we accept the biker with the intuition that if our relationships are strong on the bike, then there has been evidence of cooperative and equitable intent throughout the game thus far. Given our opinion formulation incorporates a trust framework [55], as well as a measure of effort and fairness, this decision seemed appropriate. We maintain an open-minded commitment towards acceptance, promoting stronger relationships while also minimizing the effects of selfish intentions from the incoming biker, given how cooperative the bike has been so far.

Kicking

The backbone of our agent's decision to kick someone is similarly oriented around our relationship with the biker in question. Given our BDI-based agent architecture, we implicitly retain and update information on how other agents have been performing and voting given their intentions, described in Section 5.2.1. When our opinion of an agent descends beyond a threshold, we nominate the agent for kicking as their behaviour has pointed to lack of trust, low effort/free-riding or selfish voting. While this approach is suitable in a deliberative democracy, in a dictatorship government only the dictator (see Section 5.2.3) decides when a biker is kicked off. Should we be elected the dictator of the biker, our opinions of others are affected strongly by undesirable intentions. Through messaging (see Section 5.2.2), our agent responds to bikers that deviate from the agreed upon direction, kicking them off the bike should they intend to steer of course. In a similar vein, if we discover that an agent has been free-riding (no pedalling force), then we implicitly punish them by degrading our opinion of them. Otherwise, the same kicking principle applies, nominating those with whom we have a poor relationship.

Leaving and Joining

A decision on leaving our current bike is governed by a score that compares the environmental state and our social network. The score is a weighted sum that compares our bike with other bikes, including our opinions of other agents, how many Lootboxes are in the surrounding area, the distance of the Awdi to the bike and what colour preferences agents have. This open-minded commitment approach factors in our opinions (see Section 5.2.1), aligning with our desire to uphold strong relationships without disregarding our own self-actualization needs [38], in an effort to compromise between trust and our well being.

While our agent is off a bike, it keeps track of how many rounds it remains in limbo as well as the response in acceptance from other bikes. This information is useful for our agent when making future leaving decisions, as a leaving 'risk' score is updated to analyze if leaving has proven worthy or an efficient process that does not burden the agent's energy or relationships significantly. This feedback is another example of how our agent uses prior information to come to a conclusion.

5.2.6 Lootbox Voting and Direction

Prior to pedalling decisions, bikers must nominate and vote for, assuming it is safe to do so, a Lootbox to travel towards. Bikers nominate Lootboxes, which are then voted for to find a majority preference. Cases with leadership or dictatorship governance are outlined later in this section. Our agent's Lootbox nomination strategy follows simple-minded commitment, centred around the desire to maximize accumulation of points. If a Lootbox exists of our preferred colour, we check if we can reach it given our energy circumstance. If we cannot reach it, we nominate the closest reachable box to it, otherwise if no box exists of our colour, we nominate the nearest Lootbox in a greedy effort to gather energy.

Following nomination, voting is again called to decide a nominated target Lootbox, and our agent adopts another strategy for this decision making. Trivially, should our nominated box have a majority nomination, then we follow through and vote for our nomination. Otherwise, our agent will assign a score to every nominated Lootbox. If the box in question is not reachable by our agent, or the box is not close to a box of our preferred colour, then no score is assigned to it. Aligning with our BDI application, boxes remaining in nomination are given a score based on our desire-intention structure named 'Cube Score'. This score, shown graphically in Figure 5.4 is a trilinear interpolation of our relative energy, points and relationship with the agent in comparison. The position we lie in this space is therefore governed by our BDI feedback loops, which updates our opinions of other agents and relative state. Our agent would then have more intent to support agents that have proven themselves fair and equitable, without disregarding our own needs. This strategy corresponds with the canons of distributive justice, considering votes as equals, by needs and regarding contributions and effort.

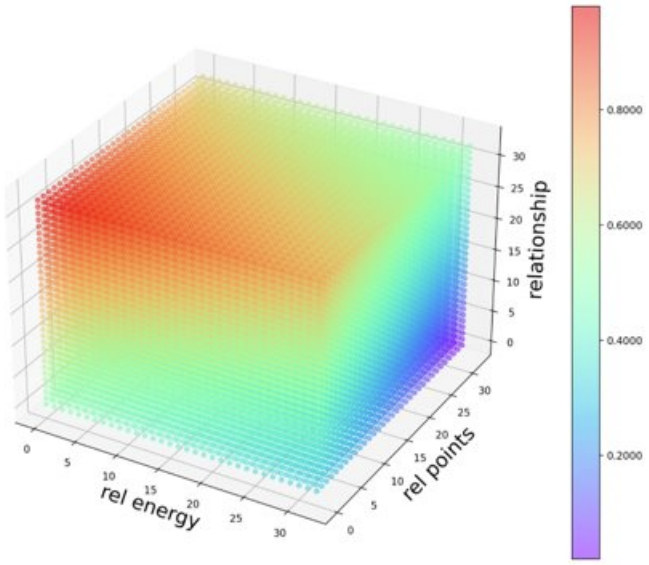


Figure 5.4: Cube Score

When the shift of governance results in our leadership (leadership democracy), a similar mechanism to Loot Distribution (see Section 5.2.4) is enforced. In light of the canons of distributive justice mentioned before, we again leverage the state of the environment and dynamic relationship information for a decision in direction. Analogous to Lootbox distribution, a decision between two bounds is made, one of a purely selfish direction decision corresponding to our preferred Lootbox, and one that is purely selfless, where the other agent's preferences are taken into most consideration. Interpolation between the bounds depends on our agent's cube score metric with others (Figure 5.4) and therefore intrinsically depends on our relative state and relationship. Every agent's decision is weighted by their corresponding score, prior to a final decision being made through weighted majority voting.

In a dictatorship government, should we be elected as a dictator, a decision is made by expecting that other agents would have blind intent to, if they could, vote to travel to their Lootbox colour. Our agent's strategy in this case is then to consider these preferences and decide a direction based on the environment and our relationships. While this seems similar to leadership democracy, it differs in that agent's true votes are not considered, and our opinions of others, as a dictator, are much more punishing should others misbehave or not conform to absolute decisions. Our BDI feedback loops and messaging therefore enables generous and strict

dictator personalities. Expecting other agents will want to go to their preferred box colour, we take the nearest reachable box of their colour and assign a score to it as shown in Equation 5.2

$$\frac{D}{n} * \frac{e}{E} * (C_i + \sum_j d_j c_j) \tag{5.2}$$

where:

D	= Number of reachable Lootboxes around target box
n	= Number of bikers on our bike
e	= Energy of the target box
E	= Our current energy level
$(C_i + \sum_j d_j c_j)$	= Cube Score of agent

This score leverages the complexity of our cube score (see Figure 5.4), highlighting our dictator opinion of them, as well as the distance to the box and density of boxes around it. This strategy adopts the canons of distributive justice, regarding self-actualization needs as equals with respect to contribution. Nevertheless, through our opinion formulation and its dynamic response while we are dictator inherently portrays the shift in governance and personality, such as dictator kicking decisions discussed in Section 5.2.5.

Following nomination and voting, depending on governance our biker will pedal and or steer to the voted Lootbox direction. These pedalling forces are calculated in proportion to our current energy, aligning with any calculations done that determined if we were able to get to the Lootbox in question.

5.3 Analysis of Experiment Performance

Experimental results were obtained across multiple simulations to analyse the performance of our agent self-organising with itself. Performance results from the Base Agent, implemented by the Infrastructure team, were also obtained for comparison. Furthermore, the results obtained confirm the intended behaviour of our agent through multiple metrics.

Figure 5.5 shows the performance of 56 versions of our agent self-organising across multiple rounds where past experiences (memory) carry over. It is noticeable that as the number of rounds increases, the survival rate of our agent increases. At the beginning, the agent seems to survive up to 40 iterations (seen in rounds 1-5) and gradually increases its life expectancy (rounds 6-7) to around 70 iterations. As the agent reaches round 9 (dark green) and round 10 (purple), the life expectancy of our agent reaches 100 iterations (maximum life expectancy).

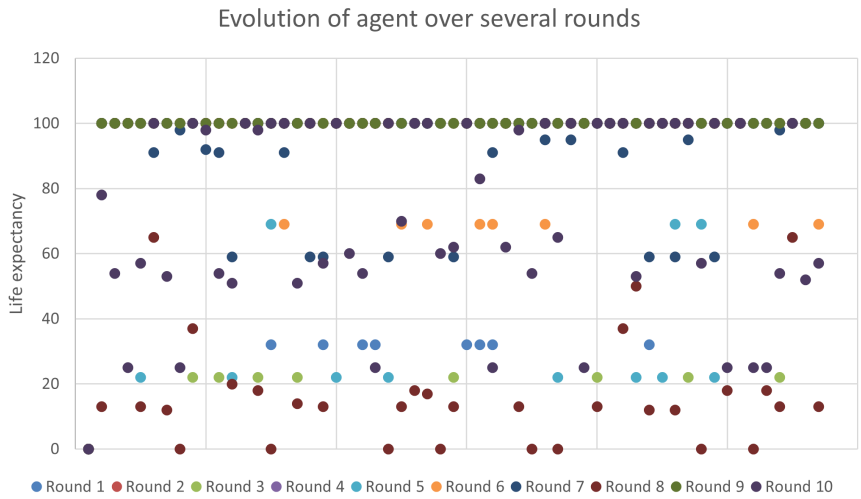


Figure 5.5: Performance of Agent 1 across 10 rounds compared with BaseBiker

The average life expectancy (Figure 5.6) of our agent and the 'BaseBiker' agent across 10 different simulations, each with 10 rounds and 100 iterations was obtained. It is noted that the initial average life expectancy of our

agent is lower than the 'BaseBiker' agent however, it gradually increases as it learns and forms its opinions across multiple rounds, leading to a more developed agent with a higher life expectancy. The 'BaseBiker' generally maintains the same life expectancy which is expected since its strategy does not change across multiple rounds.

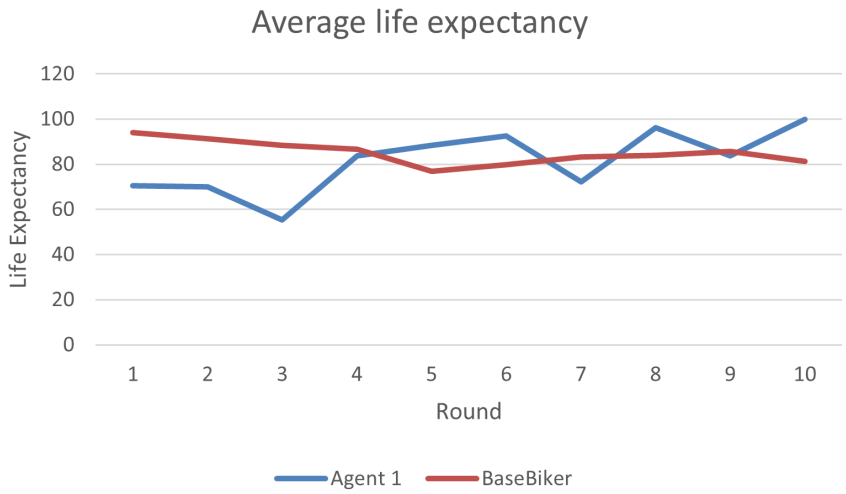


Figure 5.6: Average Life expectancy across 10 simulations compared with BaseBiker

The average energy per round (Figure 5.7) was also obtained for both our agent and the 'BaseBiker' with the same number of rounds & iterations as before. It is noted that our agent has roughly 0.1 less energy compared to the 'BaseBiker' across all iterations. The 'BaseBiker' also maintains its energy across multiple rounds whereas our agent very slowly gradually decreases its average energy level per round as the number of rounds increase. This behaviour could be an evolution of our agent's opinion of its peers (which are also our agent). The agent's trust, fairness and effort opinion metrics of its peers have been potentially decreasing in cases and thus our agent aims to compensate deteriorated relationships with more pedalling effort.

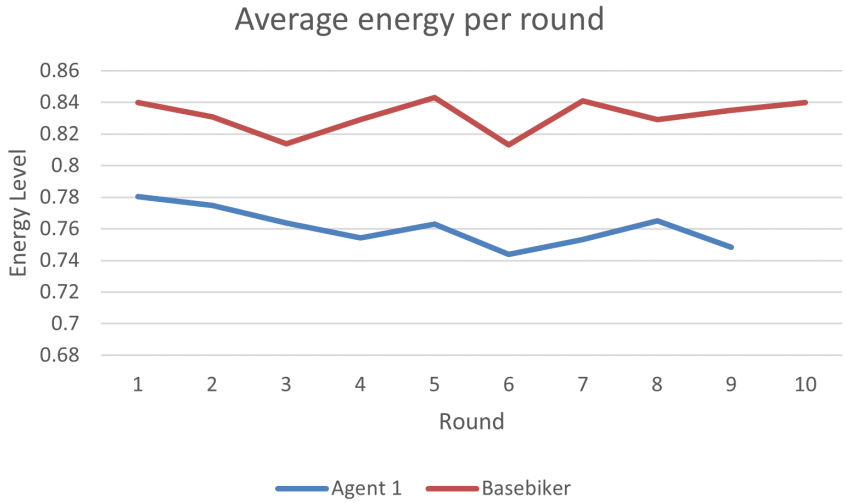


Figure 5.7: Average energy across 10 simulations compared with BaseBiker

As seen in Figure 5.8 our agent aims at maximizing its points however, across multiple rounds the agent seems to prioritize gaining points less. This could confirm the previous hypothesis that the agent is compensating for slightly deteriorating relationships and is aiming to work as a collective. However, it can be noted that in comparison to the base biker, the performance of our agent in obtaining points is slightly better. This can be owed to the agent's main desires to maximize points and energy -unlike the 'BaseBiker' which aims to go to *any* nearest box no matter the colour.

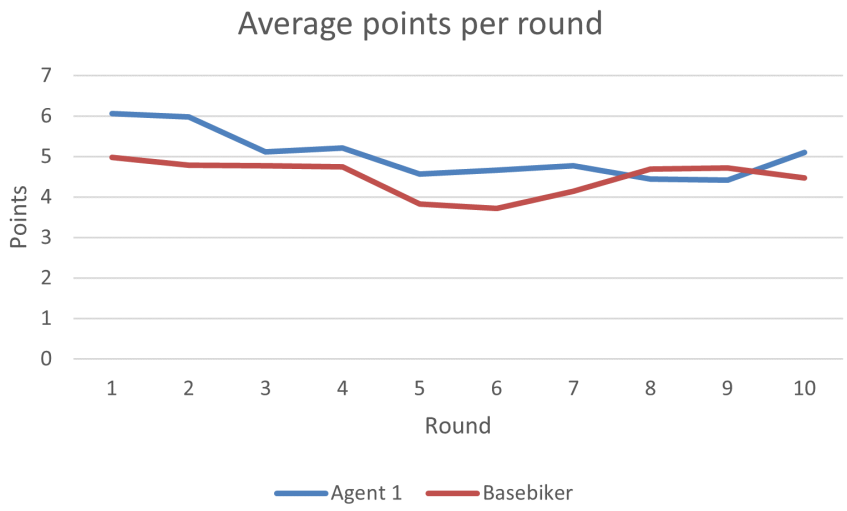


Figure 5.8: Average points across 10 simulations

5.4 Future work

Our opinion formulation and feedback loops that compose much of our agents decision making was one of the key distinguishing factors in collective action challenges compared to other agents. While we are satisfied with our agents performance compared to BaseBiker as discussed in the previous section, we recognise room for improvements in our agents strategies ranging in movement, allocation and interactions. Additionally, we considered refinement to opinion formulation itself in order to consider further specific scenarios throughout the game, a necessity if other agents were given opportunity to do so.

One potential improvement we considered was a revision in strategic foresight in regard to the credibility of others. Our trust framework (see Section 5.2.1) that formulates overall opinion relies on the recommendations of peers (as well as direct experiences) through broadcasting in order to construct the decision to trust an external agent. Agents with different relationship strategies, particularly those with heavy emphasis on social recognition and approval [10] could naivly behave credulously, which may result in there recommendations being grossly over-exaggerated. To prevent our agent suffering from this, a feedback loop could be constructed which punishes the recommender's trust/reputation should we deem it inaccurate. A similar approach can also be applied to agents that, through messaging, may lie about there state. This review and punishment loop would refine our agent's trust framework with crucial impacts to collective action.

Our trust score fundamentally observes other agent's deviation in pedalling/steering to the intended direction. In reality, scenarios could come about that force agents to deviate for the sake of survival, such as avoiding the 'Awdi' or our intended Lootbox disappearing. Our agent should consider these scenarios to avoid misleadingly punishing others that deviated for the collective good. When predicting agent deviations based on color preference and initial trust, these scenarios should be considered perhaps by introducing a dampening factor to the equation. We should also consider whether or not deviations were favoured to specific agent's preferred color, avoiding unfairly judging agents that most likely did not contribute to the deviation.

Our strategy of pedalling/steering force could be improved by leveraging more information about the environment/relationships through our feedback loops. Currently, our forces are calculated in proportion to our energy with no consideration to slight deviations. Given our relationship-centric model where our actions are governed by our opinions, it would be interesting to incorporate it into our force decisions. If we have generally low opinions of those on our bike, we may attempt to remedy our relationship if they have the same color preference to us by pedalling harder. We consider if Lootboxes are reachable based on our current implementation, therefore a lower and upper bound on force could be constructed and evaluated by our agent.

5.5 Project Management

Efficient project management relied on role allocation and an effective workflow pipeline. This section describes our management process, emphasizing regular communication and progress tracking which contributed to our success.

5.5.1 Team Distribution

Our team's structure comprised of members dedicated for infrastructure, experimentation and agent development:

Infrastructure: One member of our team worked with the common cohort infrastructure team. They focused on development and feedback of fundamental code that our agent interacts with, such as the physics and voting mechanisms.

Experimentation: Two members of our team worked with the cohort experimentation team. They worked in the exploration and experimentation of diverse agent decision-making and voting strategies. Analysis of certain experimentation on our agent was discussed in Section 5.3.

Agent Development: The remainder of the team had primary responsibility for the implementation and development of our team's agent code, with weighting on coding or report and presentation. Agent development worked on strategy research and development across different aspects of the game, including social connectivity, Lootbox decision making and bike membership, aligning with our chosen BDI-based model.

5.5.2 Workflow Pipeline

We aimed to meet as whole at least once a week to align on strategies and discuss thoughts and progress. These sessions served for in-depth discussions regarding the implementation of new strategies and the improvement of existing ones.

Each meeting enabled role allocation in effort to approach key milestones, such as the strategies for each action in agent development. This allowed for a collective understanding of each team and individuals contributions, ensuring clarity throughout the project. The use of messenger applications across different teams promoted easy organisation of meetings, sharing of ideas and management of deadlines.

Our communication methods provided a space to share diagrams or graphs created during discussions, as well as alignment on deliverable implementations. The use of a shared repository facilitated easy contribution and implementation of methods for our agent through branch management and conflict handling.

5.5.3 Conclusion

The integration of this workflow pipeline, combined with regular meetings, role allocation discussions, continuous communication channels, and collaborative tools for visual representation and discussions, played a crucial role in our team's success. This approach allowed for an efficient problem-solving environment both within and across sub-teams, ultimately contributing to our project goals. Fundamentally, our agent was compatible was the underlying infrastructure, implementing its unique strategies in SOMAS World and proving successful in experimentation.

Chapter 6: Team 2

6.1 Introduction

Collective action problems are characterized by the involvement of multiple agents, each with individual goals that may conflict with a shared objective. This leads to situations where individual rationality leads to collective irrationality [50]. SOMAS World presents a collective action problem where agents must survive for as long as possible by using their energy to loot Lootboxes.

We explore social capital to address this collective action problem. At selected events, agents update metrics over the forms of social capital, which are then used for key decisions in SOMAS World. Social capital frameworks simplify decision-making and have led to "higher long-term satisfaction and utility" in other collective action problems [53].

6.2 Background

Social capital enables participants to collaborate to solve collective action problems and achieve shared objectives [1, 50]. We explore the main aspects of Social Capital in our agent model: Trustworthiness, Social Networks, Institutions, and Forgiveness.

1. **Trustworthiness**: Trustworthiness refers to the reliability and integrity of agents within a system; highly trustworthy agents exhibit behaviours that contribute to our objective. Our model updates trustworthiness from reputation events, primarily quantifying the extent to which an agent's actions align with our interest.

2. **Social Networks**: Social network, another pillar of Social Capital, represents the relationships and connections among agents. These networks facilitate the flow of information and resources, crucial for collective decision-making and achieving common goals [1]. In our model, the network score of an agent increases with sustained cooperative interactions, reflecting the depth and strength of their relationships within the system.

3. **Institutions**: Institutions refer to the established rules and norms governing agent interactions [50, 58]. Pitt's work on electronic institutions in multi-agent systems provides a framework for understanding how rules and decision-making processes can be structured to promote cooperation and collective action [58]. In our model, the institutional score of an agent is influenced by their adherence to these established norms and their role within the system's hierarchy.

4. **Forgiveness**: From [77], forgiveness in the realm of Social Capital involves the capacity of a system to facilitate recovery and reintegration following breaches of trust or cooperation. This aspect of Social Capital is particularly significant in maintaining the adaptability and resilience of social networks. In environments where agents continuously interact and are expected to cooperate, the ability to forgive and readjust relationships is vital despite occasional conflicts or errors. Within our model, forgiveness serves as a dynamic mechanism for adjusting trust and reputation levels as suggested in [53]. This allows the system to sustain cohesion and recover from disruptions caused by individual agent actions, thereby enhancing the overall stability and effectiveness of the multi-agent system.

These forms of social capital foster *trust* for successful collective action [53]. By incorporating these principles, our model aims to foster an environment where agents effectively self-organize to address the collective action problem SOMAS World presents.

6.3 Implementation and Design Decisions

6.3.1 Overview

Social Capital is used to make decisions during voting and messaging events. Throughout the game, an agent keeps track and dynamically updates Social Capital based on events with various weightings. Fig. 6.1 illustrates our strategy.

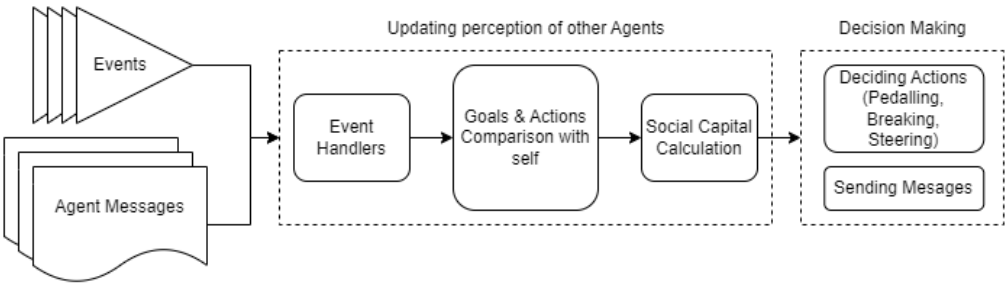


Figure 6.1: Our Agent Strategy

6.3.2 Social Capital

Each agent holds a social capital value of others; $SC^t_{i,j} \in [0,1]$ represents the social capital value agent i assigns for agent j in a round t. Here, $SC_{i,j}$ is the weighted sum of institution, reputation, and social network scores: $I_{i,j} \in [0,1]$, $R_{i,j} \in [0,1]$ and $N_{i,j} \in [0,1]$ respectively (see Eq. 6.1).

$$SC^t_{i,j} = w_I \cdot I_{i,j} + w_R \cdot R_{i,j} + w_N \cdot N_{i,j} \tag{6.1}$$

Values $I_{i,j}$, $R_{i,j}$ and $N_{i,j}$ are updated by events in SOMAS World, and $SC^t_{i,j}$ is updated at the end of every round. We initialize $SC^0_{i,j}$ to 0.5, assuming that agents are neutral about other agents they have not interacted with. Differences in $SC^t_{i,j}$ with $SC^{t-1}_{i,j}$ may be forgiven (see Sec. 6.3.6). Within SOMAS World, $SC^t_{i,:}$ values are used by an agent i in the following decision events. We posit that these rules allow our agents to reorganize into bikes with other agents they *trust*.

- `DecideAction`: Agent i will change bikes if the average social capital of its fellow bikers is below a set threshold. Otherwise, it pedals.
- `VoteLeader` or `VoteDictator`: Agent i votes for itself another agent j with with maximum $SC^t_{i,j}$ with equal weighting.
- `VoteForKickout` or `DecideKickout`: Agent i votes for another biker j with the minimum $SC^t_{i,j}$ to be kicked out.
- `DecideJoining`: Agent i accepts agent j's request to join its Mega-Bike if it's $SC^t_{i,j}$ is larger than a set threshold.
- `ChangeBike`: If Agent i decides to change bikes, it tries to change bikes to one with agents with the highest average social capital values.
- `DecideAllocation`, `DecideDictatorAllocation` or `DecideWeights`: Agent i votes for loot distributions and vote weightings based on $SC^t_{i,j}$. Agents reward other agents it *trusts* more.

Fig. 6.2 shows the changes in social capital in different iterations of the game depending on the actions of agents. The size of the array of social capital scores increases as new players join the bike.

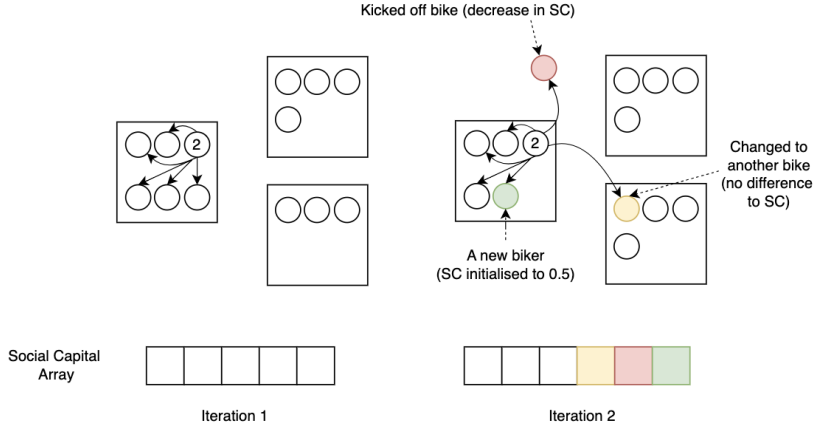


Figure 6.2: Changes in Social Capital Over Different Iterations

6.3.3 Institution

We increment and decrement $N_{i,j}$ based on institutional events. The events in SOMAS World that impact $N_{i,j}$ are the following,

1. **Accept Biker**: When agent j is accepted onto a bike with agent i, $N_{i,j}$ is incremented by a small value. We keep the increase small to allow agents to quickly discern if the accepted agent has aligned goals. If not, the accepted agent would be voted off the bike.
2. **KickOff Biker**: In contrast, $N_{i,j}$ is penalized by a large amount if agent j is voted off the bike. The large penalty avoids "bike jumping"; when tuning this penalty, values lower than the reward of being accepted onto a bike caused agents to change bikes repeatedly. This, in turn, inflated the agent's social capital values.
3. **Voted Roles**: We increment $N_{i,j}$ if agent j is voted to be a dictator or leader. If an agent gets voted as a dictator or leader, it can be perceived that the agent's goals are well aligned with others, thereby garnering more votes from other bikers.
4. **Vote Participation**: Each agent will be affected by voting results; hence, they can modify the operational rules of SOMAS World. This ties into Collective Choice Arrangement [57]. Since the overall decided Lootbox is heavily influenced by voting (for both the decided institutionalized power and Lootbox), each agent's participation is crucial.

6.3.4 Reputation

$R_{i,j}$ is a measure of how close agent i's goals align with agent j. We update this value in a `HandleForcesMessage` event by comparing the forces another agent j intends to exercise versus those in agent i's best interest (towards a Lootbox of its desired colour).

We note that this strategy assumes all agents are truthful. In a SOMAS World with byzantine/lying agents, reputation values may be inaccurate in measuring how an agent's goals align with every other agent, and it's weight w_R might need to be reduced.

6.3.5 Network

To model relationships between other agents, we increment $N_{i,j}$ every round agent j has been on a bike with agent i, whenever agent j has shared messages, and whenever agent j has collected a Lootbox with agent i. These metrics favour bikers with whom it has spent more rounds and indicate *trust* if they have not been kicked yet.

6.3.6 Forgiveness

Forgiveness was modelled as tokens to reconcile relationships between the team's agent and other agents. If $SC_{i,j}^{t+1} < SC_{i,j}^{t}$, a forgiveness factor ($f \in [0,1]$) would be utilized to dampen the decrease in $SC_{i,j}^{t+1}$. Agent i can only forgive agent j for up to three consecutive rounds (Eq. 6.2). This policy punishes other agents whose Lootbox goals do not align consistently and facilitates recovery of $SC_{i,j}^{t}$ values.

$$SC_{i,j}^{t+1} = SC_{i,j}^{t+1} + (SC_{i,j}^{t} - SC_{i,j}^{t+1})f, \qquad f > 0 \text{ only for 3 rounds consecutively} \tag{6.2}$$

6.4 Observations

6.4.1 Clique Formation

Taking a higher-level analysis of our experimental results, we noticed that our agent, governed by Social Capital principles, tends to act favourably and form a clique. The main difference between Social Capital and favour is that while social capital is typically interpreted as a more impartial measure of how an agent operates within an environment, favour takes a far more selfish approach to measuring how useful other agents are to our agent. For example, our implementation of reputations measures the alignment of interest between our and another agent's objective. This indirectly encourages the formation of cliques. We will dive into our observation of clique formation and its effects on an agent's performance.

Events of agent interactions in SOMAS World are used to calculate social capital. Trustworthiness, a central pillar of social capital, measures how much effort an agent puts into pedalling toward our desired Lootbox, the closest Lootbox of our desired colour. This encourages the formation of cliques as agents with the same

colour would have high trustworthiness scores of one another. With a high social capital score, our agent would cast a higher weighting during the voting stage, which leads to a collective decision of kicking agents off or allowing agents in. Experiments showed that the frequency of bike hopping reduces significantly after iterations, indicating that agents can self-organize into cliques with common interests. Agents decide to switch bikes when it does not *trust* other agents on a bike.

Alternatively, network values, $N_{i,j}$ favoured bikers agent i has spent more rounds; the longer an agent has been on a bike with another agent, the stronger their social ties are. However, this enabled the establishment of cliques with common interests. In our case, this means the agent will favour agents it has spent a longer time with (which is some indication of trust if they haven't been kicked) and will allocate resources and pedal more for its "friends" as it establishes an in-group of a longer standing member of the bike. Experiments showed that agents with common Lootbox interests organized themselves onto common bikes.

6.4.2 Effects on Agent's Performance

To investigate the effects of social capital on an agent's performance, we initialized a heterogeneous population of base agents and social capital agents. The base agent acts purely on pedalling to the nearest Lootbox with an equal split of resources. This provides a control to compare our Social Capital agent's decision. We have observed that social capital-based decision-making and clique formation improve the overall lifespan and average score of agents in the clique.

We completed experiments on comparing our agent's performance with the Base Agent, and concluded that our agent survives for longer on average regardless of the governance selected. In Fig. 6.3, 6.4 and 6.5, we show both the averages of all agents throughout the simulation, as well as the behaviours of all our agents throughout.

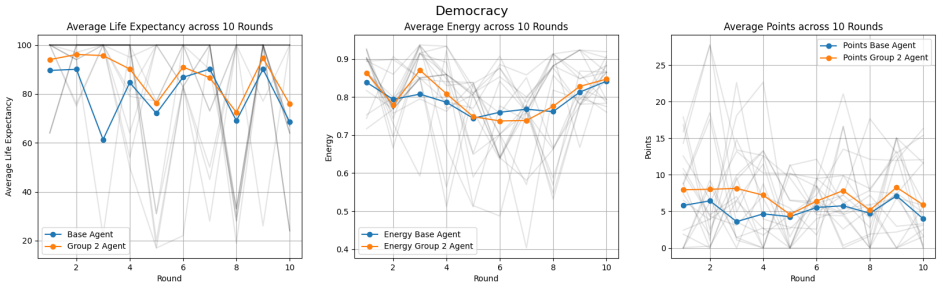


Figure 6.3: Agent Statistics on Democracy

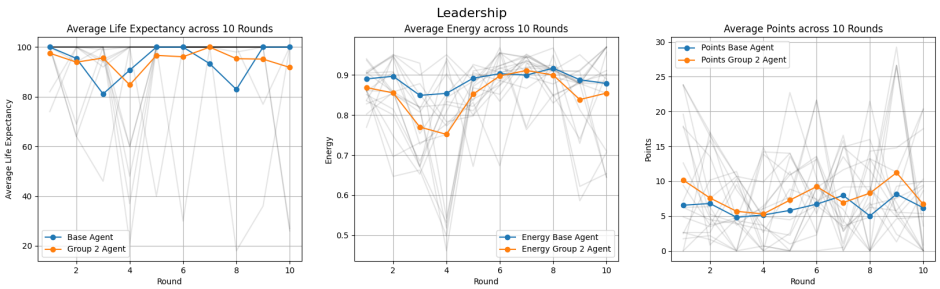


Figure 6.4: Agent Statistics on Leadership

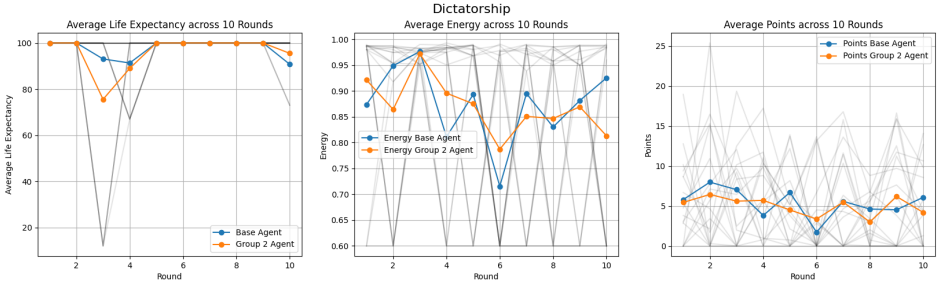


Figure 6.5: Agent Statistics on Dictatorship

Qualitative Observations

We observed that our social capital agent needed information on other bikers during the initial phase due to the need for more interaction between the agents. This leads to similar pedalling decisions with the base agent, which conforms to pedalling to the nearest Lootbox. If pedalling to the nearest Lootbox is not in our agent's best interest, bike hopping is performed to find a bike with a similar interest.

After iterations, agents with common interests were observed to organize onto common bikes, and the frequency of bike switching was greatly reduced. Agents with aligned goals will likely Lootboxes more efficiently, increasing overall energy and points.

Quantitative Observations

Fig. 6.6 shows the life expectancy for each team's agent. Our agent achieves the highest life expectancy of 89 rounds, +17% higher than BaseBiker - our baseline. This indicates that our social capital framework enables agents to self-organize to optimize life expectancy in SOMAS World.

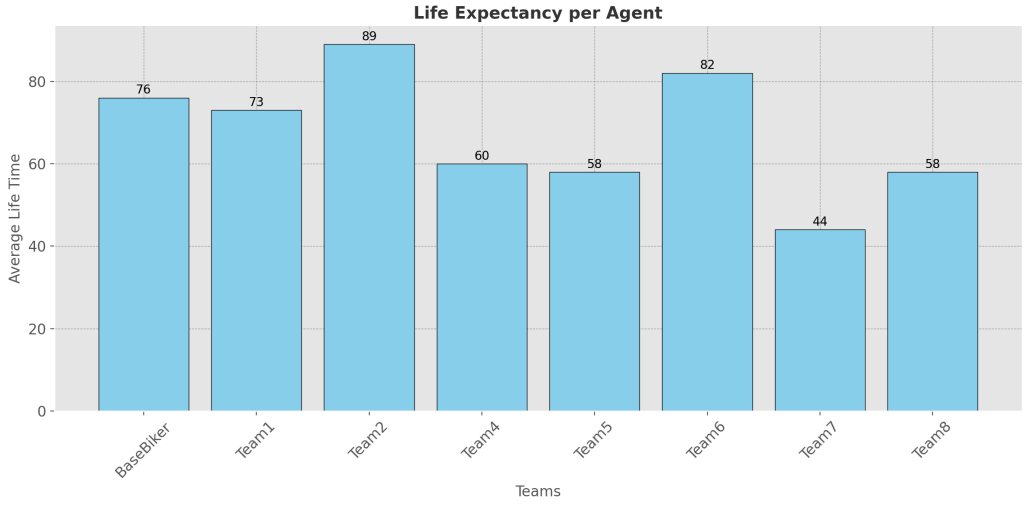


Figure 6.6: Life Expectancy per Agent

6.5 Future Work

6.5.1 Tuning

- Social Capital Weights: Currently, the weights for each component are not set to optimal value. With extensive testing/tuning a different set of weights may have been chosen that can further optimise our agent for SOMAS World. The sum of weightings would still sum to 1, but the relative importance of each component can be adjusted.
- Event Weights: The effect on the social capital of each event can be fine-tuned. Currently, we set the value arbitrarily with our subjective importance of each event. However, event value can be improved to better align with the relative importance of events.

A challenge associated with tuning is requiring a set range of numbers to enumerate during experimentation. An initial estimation can be made by analysing the agent's strategy to address this challenge. This analysis aims to determine which components are more crucial relative to all the parameters, which can provide a starting point for tuning.

6.5.2 Experimentation

- Population of Agents: Currently, our analysis was performed on a heterogeneous mix of agents with different strategies. Performing experiments with only one other agent of a different strategy would allow us to draw richer insights regarding the drawbacks and benefits of our social capital framework in optimizing life expectancy.
- Weights for each Social Capital Pillar: Each of the social capital pillars has a different weighting contributing to the social capital score. The score directly affects the decisions made by the agent when

deciding whether to stay on the same bike, join a new bike or kick another agent off the current bike. Therefore, by varying these weights, the agent's behaviour can drastically change. Depending on another agent's strategy, a set of different weightings may yield better outcomes within the SOMAS World game. Therefore, exploring the relationship between different strategies and social capital weights would provide some clarity on the effectiveness of social capital in different situations.

- Forgiveness formulation: Presently, forgiveness is modelled using Eq. 6.2 and is applied for all agents regardless of their distribution of social capital pillar scores or strategies. Since forgiveness is applied to the overall social capital score, it does not take into account which pillar caused the decrease in the social capital score. Incorporating forgiveness in the social capital pillar level can emphasise a decrease in score for a specific pillar. Depending on whether the pillar is heavily weighted, it can therefore have a bigger influence on the overall social capital score.

6.5.3 Communication Strategy

Our approach to perceiving agent communication allows for future implementations such as lying detection. Based on the Social Capital of another agent and the overall bike movement, our agent could decide whether we believe in the message they conveyed. This would have required extensive testing of the messaging infrastructure and agents from other groups to be well implemented. As mentioned previously, our current agent assumes that the majority of communication in SOMAS World is truthful. In a world full of deceptive agents, our agent may be misled and assign a high social capital to a deceptive agent.

Our agent always communicates truthful information regarding its actions, decisions, and aims (pedalling forces, preferred Lootbox, etc). This is to avoid being caught lying by other agents and losing credibility. We believe that a more truthful agents achieves better results as it promotes honesty and collaboration within the game, both of which are beneficial for collective action problems.

Chapter 7: Team 3

7.1 Introduction

Team 3 conducted an empirical study to examine the collaborative dynamics and interactions among agents possessing distinct personality traits within the SOMAS World environment. The investigation focused on characterizing the behaviors of agents with selfish, communist, and smart personalities, aiming to gain insights into their collaborative patterns and engagement with other agents within the simulation framework. This research contributes to a deeper understanding of how diverse agent personalities influence collaborative efforts in SOMAS World, shedding light on the implications for multi-agent systems and cooperative environments.

Section 7.2 introduces our agents' strategies and common functions that offer references for our agents to make decisions. Section 7.3 details the agents' decision-making processes, covering leader selection, bike choices, member acceptance, Lootbox preferences, voting, pedalling, energy allocation, and messaging. Section 7.4 presents the results and analysis of our experiments, Section 7.5 encapsulates our conclusions, and Section 7.6 outlines directions for future work. Finally, Section 7.7 provides an overview of the project management diary.

7.2 Overall Agent Strategy

Our agents operate as teleo-reactive programs, which receiving and collecting information from their environment and responding in specific situations. The guiding philosophy for all decision-making functions revolves around considering three key elements of input: the SOMAS Physic World, reputation judgments for all agents, and the satisfaction of current cooperative peers (overall assessment of fellow agents on the same Mega-Bike).

In the context of the Prisoner's Dilemma, our strategic approach is geared towards fostering a positive outcome where all agents cooperate to the best of their abilities. However, in instances where our agents perceive the team dynamics as unfavorable, marked by low satisfaction levels, a more cautious approach is adopted to avoid falling into the negative consequences of potential betrayal.

When faced with a lack of sufficient knowledge about the behavior of other agents, our default strategy is to lean towards cooperation, investing maximum effort, and distributing resources equally. As our agents accumulate experience, we employ both the Ramirez-Cano-Pitt model and the Hegselmann-Krause model to evaluate the decisions of other agents through the lens of reputation judgment. In accordance with the social comparison theory, our inclination is to align more closely with decisions made by agents possessing a high reputation.

Over the course of several cooperative rounds, our agents form a judgment on the satisfaction levels within the team. This assessment influences our decisions on the extent of cooperation with the team, ranging from being a maximum force contributor to a free-rider or even leave the bike. Additionally, we consider whether to convey accurate information or misrepresent our actions and thoughts to our fellow agents.

7.2.1 Reputation Function

A **reputation** system was developed for common use to demonstrate how the agent trusts and forgives others in such a multi-agent system. The system generates a **reputation map** designed to evaluate individual agents within the societal framework, assimilating pertinent information (shown in Table 7.1) for informed decision-making and facilitating interactions among agents to optimize our collective interests [36]. The reputation map performs as a common-use factor for different decision-making sections. The map contributes to *opinion formation* and *reputation calculation*.

Opinion Formation

The reputation map is a dynamic evolving specification, formulated during the design time and subject to run-time modifications. It encapsulates the performance of each individual agent on the bike, incorporating both cognitive and economic dimensions to assess and quantify the level of trustworthiness attributed to other agents.

Features	Description
recentContribution	Energy contribution of agents in the current turn
historyContribution	Energy contribution of agents in the last two turns
opinionSimilarity	The similarity of proposals
isSameColor	Is the color of the proposed target Lootbox same as our desired
lootBoxGet	The number of Lootboxes gained in the past of our agent
energyGain	The amount of energy gained in the past of our agent
energyRemain	The amount of energy remained after the current turn
recentGetEnergy	If the agent gained energy in the most recent turn
_lastPedal	The number of pedalling contributed in the last turn
_pedalCnt	The total number of pedalling contributed
_lastEnergyLevel	Energy level at the end of the last turn of other agents on the bike
_recentEnergyGain	Energy gained in the most recent turn of other agents on the bike
_energyReceivedCnt	The amount of energy received in the last turn
_lootBoxGetCnt	The number of Lootboxes gained in the past of other agents in the past

Table 7.1: Features Involved in the Reputation System

Following every turn, the society disseminates information regarding energy and scores, facilitating the iterative refinement of our reputation map for each individual agent. In order to optimize interaction efficiency with other agents, our agent selectively retains contribution factors for a limited span of two turns, thereby affording a degree of forgiveness towards agents exhibiting predispositions toward inactivity within the societal context.

The reputation system acts as a central mechanism to systematically aggregate and refine evolving knowledge in each subsequence cycle. Its primary function is to enable agents to extract and leverage accumulated information to facilitate informed decision-making in real-time scenarios. By integrating historical and contemporary data, the system creates a dynamic library of advanced intelligence. This iterative process allows for differential adaptation to different environmental stimuli and allows actors to identify optimal courses of action. Thus, reputation systems improve the cognitive abilities of agents and create an environment in which more effective informed and contextual decisions can be made.

Pseudocode:

Listing 7.1: Code Example

```
func updateScore (biker objects.IBaseBiker, preferredColor utils.Colour):

    Update _lastEnergyLevel
    Update _pedalCnt

    Update recentContribution
    Update historyContribution
    Update energyRemain
    Update energyGain
    Update lootBoxGet

   Distinguish if the target color is our preferred
```

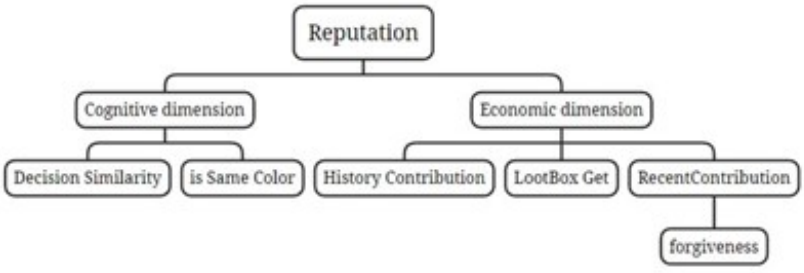


Figure 7.1: Reputation Features

Reputation Calculation

The computation of reputation involves the assignment of weighted values to the aggregate score, constituting the summation of individual factor scores within the reputation map. These weights are contingent upon the specific tasks at hand and the distinctive characteristics inherent to each agent. This deterministic weighting mechanism allows for a tailored evaluation that reflects the nuanced relevance of each factor in alignment with the unique attributes and responsibilities associated with diverse tasks and agent profiles. The deliberation of weights ensures a nuanced and context-sensitive assessment, thereby enhancing the precision and adaptability of the reputation calculation process across varying scenarios.

$$reputation = weight * \sum_{n=eachfeature} Score_{eachfeature} \tag{7.1}$$

7.2.2 Satisfaction

The satisfaction means how we satisfied with recent allocation and how fair it is. We will consider reputation rank of all fellows and the energy they gain in this allocation process following with these steps:

1. Obtain the allocation information and calculate the reputation for each agent.

2. Sort the agents by their reputation and analyze their resource gain in this order.

3. Count the number of inverse number pairs[1] in the agent resource gain sequence after sorting.

4. Normalize the count of inverse number pairs:

$$n = Number\ of\ fellows \tag{7.2}$$

$$count = Number\ of\ inverse\ pairs \tag{7.3}$$

$$Satisfaction = 1\text{-}\frac{2 * counter}{n * (n\text{-}1)} \tag{7.4}$$

Since the count's minimum value is 0 (for a matching order) and the maximum value is $\frac{n*(n\text{-}1)}{2}$ (for a reverse order), satisfaction ranges between 0 and 1. A satisfaction of 1 indicates that the allocation order matches the calculated reputation order perfectly. The greater the discrepancy, the lower the satisfaction, reaching 0 when the allocation order is completely opposite to the reputation order.

7.2.3 Dynamic Environment Adaptation

With the rising complexity and dynamism of automation systems, it is crucial to optimise the decision-making process of intelligent agents, considering the proportional cost of change in the system's size.

[1] An inverse pair is defined as a $pair(i, j)$ where: $0 \leq i < j < length\ of\ the\ array$, and the value at i is greater than the value at j. In arrays or programming, "inverse number pairs" refer to pairs of elements in an array where the first element is positioned after the second element and is greater than the second. For example, in the array [2, 3, 1], the inverse number pairs are (2, 1) and (3, 1). Counting the number of such pairs in an array is a common problem in computer science and algorithms, often related to the efficiency of sorting algorithms and data structures. This concept helps in understanding the orderliness of an array and the performance of algorithms.

7.3 Implementation of Agent Specific Functions

7.3.1 Governance

Basic Concepts

Governance Methods

In the MVP world, there are three primary types of political regimes (governance methods): democracy, leadership, and dictatorship. However, on our Mega-Bike, we utilize only **democracy** and **leadership** as our governance methods.

The democratic governance method on our Mega-Bike involves a **collective decision-making process**. This empowers all agents on the bike with the rights and responsibilities to independently engage in all activities. These activities include proposing initiatives (Lootboxes), voting on strategic directions, managing the energy used for pedalling, allocating new resources, and deciding on the inclusion of new members.

Governance Through Leadership

The leadership governance model mandates that the appointed leader devises strategies for the collective, thereby facilitating the effective completion of tasks by all participants on the Mega-Bike. Divergent from a dictatorship where absolute power rests with the dictator, this leadership governance is subject to **periodic review**. At the conclusion of each cycle, the performance of the leader is rigorously evaluated by all members. This assessment process is pivotal in deciding **whether the leader retains their position or steps down** for the subsequent cycle.

It is imperative to recognize that the **governance method is subject to modification via a democratic process.** Specifically, at the end of each cycle, especially following a democratically governed round or the displacement of the previous leader, a voting process is initiated. This process involves the participation of all members, who collectively decide the governance model to be adopted for the next cycle. This indicates an **inclusive decision-making approach**, where each individual's perspective from **various backgrounds** is acknowledged and valued.

In summary, the democratic governance method employed on our Mega-Bike is reminiscent of the city-state democracy that was prevalent in ancient Greece. This approach emphasizes **collective decision-making** and **shared responsibilities** among all participants.

On the other hand, the leadership governance method mirrors the presidential election system found in modern democratic nations, where a leader is chosen to strategize and guide, yet remains accountable to the group through periodic evaluations **(leader's accountability).**

Implementations

Part One: Leadership Election Process

The leader selection procedure employs a **two-round runoff voting system**, adhering to **the plurality method** principle in both stages.

In each round, the smartagent assigns a floating-point score to every agent on the bike, taking into account a range of factors.

The First Round

For each agent on the bike:

- **Cumulative Energy Contribution Over Time (History Contribution):**
  Pareto Principle, Whether its contribution significantly exceeds that of the other agents (as 20% of the agents may contribute over 80% of total contribution during pedalling)

- **Total Number of Lootboxes Collected:**
  Individual Contribution in Collective Action, the resources collected will become **common-pool resources (CPR)** for all agents
- **Agent's Target Color Alignment:**
  Assessing whether the agent's target color is the same as the smartagent's in this round, a factor in the **Cognitive Dimension** of the same belief.
- **Energy Reserve of the Agent:**
  This is crucial for **Governance**, as the leader must preserve the current political regime from being threatened, ensuring they have enough energy to lead the team (for instance, if the forward speed is too low, it increases the likelihood of being caught by Awdi).

The Second Round

For each agent on the bike:

- **Its recent energy contribution.**
  Forgiveness: By putting emphasis on recent energy contribution, forgiveness is given to those who pedal harder recently (they may not have pedaled hard in previous rounds).

Regarding the three distinct personality types of agents (selfish, smart, communist), each type applies unique weights to the evaluation metrics. Nonetheless, they all adhere to a uniform algorithm that computes a floating-point score for each candidate agent on the bike:

Algorithm: Electing a Leader

Input: Each agent (candidate) on the bike and its reputation parameters.

Output: A float64 score for each agent(candidate).

Get ID for each candidate agent.

Read its reputation map.

Voting Round 1:
for each agent in agents **do**

score1 $\leftarrow w1 \cdot \text{historyContribution} + w2 \cdot \text{lootBoxGet} + w3 \cdot \text{isSameColor} + w4 \cdot \text{energyRemain} + \text{Epsilon}$ **end for**

The following represents the allocation of weights based on the voter's personality.

Personalities	w1	w2	w3	w4
Smart	0.4	0.3	0.1	0.2
Selfish	0.1	0.2	0.5	0.2
Communist	Random Allocation of weights each round			

Figure 7.2: Allocation of weights based on the voter's different personalities.

Note:

1)Each agent's score will be normalized by the score accumulator.

2)The Epsilon is used to avoid dividing by 0.
Voting Round 2:

score2 ← recentContribution + Epsilon

No weight is put here as the score will be normalized so it makes no difference, and the formula is applied to agents of all three types of personalities.
Final Voting
We give a weight for scores1 and scores2, and derive the final score for each agent (candidate)
scores[id] ← $0.7 \cdot$ scores1[id] $+ 0.3 \cdot$ scores2[id]
Return scores (return type: $map[uuid.UUID]float64$).

The conclusive presentation of the election results implements the **Borda Count voting method**. This approach diverges from the plurality principle, which typically identifies the candidate with the highest score. Instead, the Borda Count method comprehensively lists all candidates on the bike, accompanied by their respective scores. These scores are presented in a map format, providing a detailed and transparent view of each candidate's performance in the election process

Part Two: Voting for Governance Method

In our model, we presuppose that **by default, agents favor leadership.**

We identify two specific scenarios where agents might deviate from accepting leadership (i.e., **Under these two specific scenarios, agents choose democracy**):

Scenario 1: Consistently Low Energy Contribution

This scenario typically aligns with agents exhibiting a **selfish** personality. In instances where an agent recognizes their minimal contribution, there may be a concern that under a leadership system, they might be compelled to contribute more energy or face potential repercussions. Consequently, such agents are inclined to favor democracy.

Scenario 2: Substantial Energy Reserves

This situation also tends to reflect a **selfish** personality trait. Agents with a significant amount of residual energy might be apprehensive that, under a leadership regime, they would be expected to expend more effort in pedalling. To avoid this, they might prefer a democratic system where such demands are less likely.

Algorithm for Voting for Governance Method:

Initialise average-recent-contribution := 0.0
Initialise average-contribution := 0.0
Initialise average-energyRemain := 0.0
for each agent on the bike **do**
Read each agent's reputation map based on its ID.
Accumulate the recent-contribution of an agent
Accumulate the contribution of an agent
Accumulate the energyRemain of an agent
end for

Calculate the average for these three indicators

Scenario 1: Consistently Low Energy Contribution
if agent-recent-contribution < average-recent-contribution **and** agent-contribution < average-contribution **then**
The governance method shifts to Democracy.
end if

Scenario 2: Substantial Energy Reserves
if agent-energyRemain $> 2 *$ average-energyRemain **then**
The governance method shifts to Democracy.
end if

Part Three: Decision to Displace a Leader

We operate under the assumption that agents are inherently cooperative and possess a sense of **collective responsibility**, which leads them to generally refrain from voting off a leader.

However, an agent may choose to **vote off the leader** if they perceive that the leader's decisions adversely affect their personal circumstances.

Algorithm for Voting off a Leader:

By default, Vote_off = false
Read the agent's reputation map based on its ID.
if an agent did not receive any energy in the last round **and** the leader's target color is different from the agent's **then**
Set Vote_off = true
endif

This mechanism of voting off a leader is relevant for both **selfish and smart agents**. These agent types prioritize their individual benefits and are likely to displace a leader whose decisions do not align with their personal interests or outcomes.

7.3.2 Change Bike & Joining Decision

Strategy of Change Bike

Change bike needs to take the current environment into consideration (e.g. the reputation of other agents and state of the bike) to make a decision about changing bikes. During the game process, our intelligent agent will dynamically adjust its behavior (e.g. change the bike or not) to adapt and make the optimal decisions for the current game state. In complex systems with fast, frequent, long-distance, and entangled changes, response needs to be guaranteed in order to effectively handle changes. In the 'ChangeBike' function, the decision-making process of the intelligent agent is strategically responded to the complex dynamics of its environment, intending to maximise its benefits or accomplish its goals within the game.

The implementation of the ChangeBike function is as follows:

1. The function initializes an identifier of the target Mega-Bike to record the selected final bicycle.

2. Set a variable (highestAvgScore) to track the highest average reputation score of the bike encountered.

3. The function iterates over all bikes in the game state, skipping the bike if it is already at maximum capacity.

4. Calculate the average reputation score of all agents on each bicycle that is not full. This score is derived from multiple attributes of the agents, such as similarity of beliefs, historical contributions, recent contributions, and remaining energy.

5. If the average reputation score of a bike exceeds the presently recorded highest score, update the highest score and return the bike ID with the highest average reputation score and switch the smart agent to target the bike.

Algorithm for Function ChangeBike Strategy:

highestAvgScore <- initial_score=0
targetId <- null
For each Mega-Bike in the game state:
loot :=If targetId is null< −: Set targetId to the current bike's ID
If the number of Agents in the current bike == BikersOnBike
Continue to the next loop iteration (ignore this bike)
Initialize score <-0
For each Agent in the bike:
Retrieve the agent's reputation from reputationMap
Calculate score based on the following criteria = Same color (isSameColor), Historical contribution (historyContribution), Lootbox acquired (lootBoxGet), Recent contribution, (recentContribution), Remaining energy (energyRemain)
If score > highestAvgScore)
Update highestAvgScore <- the current average score
Update targetId <- the current bike's ID
Return targetId

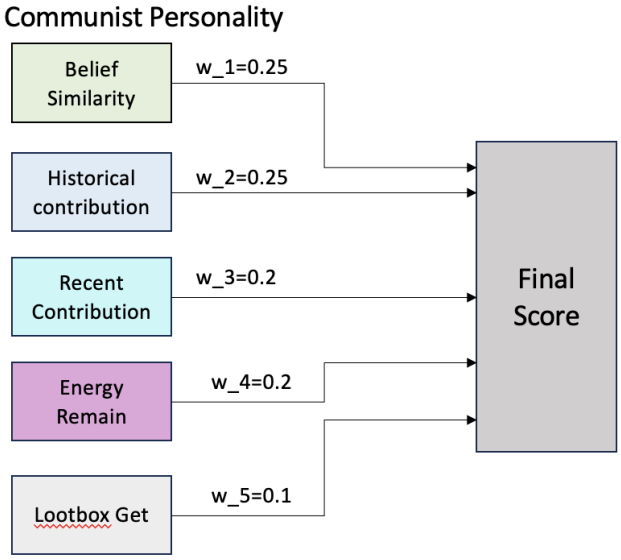


Figure 7.3: Different weight distribution of communist personalities. (Change bike)

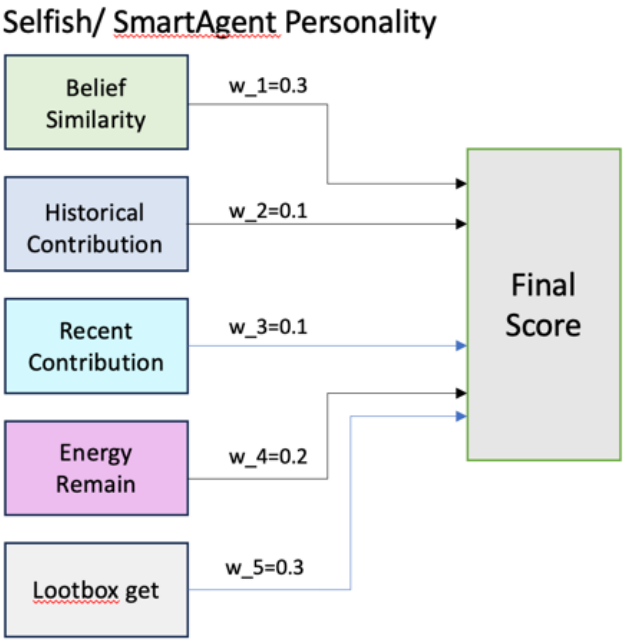


Figure 7.4: Different weight distribution of Selfish/SmartAgent personalities. (Change bike)

About the score calculation, we take the same weights to each attribute with communist personality. Under the frameworks of selfish and team-oriented personalities, we prioritize the decision to target the same color Lootbox and the capability to acquire it, assigning high weights to these factors. Conversely, the weights for the other three characteristics are calibrated to medium and low levels for scoring purposes. As the preferred colour is the same, this can benefit the entire bike. Additionally, the acquisition of the Lootbox demonstrates the agent's resource-gathering skills and their performance in the game. However, the other three attribute indicators have less significance. In this case, our emphasis is primarily on immediate benefits rather than long-term connections or team cohesion.

The strategy of changing bikes embodies analysing the current dynamic environmental state and effective strategic decision-making based on agent reputation in the automation and self-organising systems theory.

Whether to allow other agents to join the bike

The DecideJoining function determines whether to allow other agents to join our bike, depending on the aggregate reputation score of each applicant agent. This scoring system takes into account multiple factors, including compatibility of belief preference, historical contributions, recent contributions, and energy surplus. Subsequently, candidate agents are ranked based on their performance on the above attributes to determine whether they can join our bike.

In the function "DecideJoining", the overall rating of agents is similar to the ranking system in Borda Count, but it is based on predefined criteria (such as historical contribution, energy surplus, etc.) rather than the subjective ranking of voters. Agent score ranking is similar to the Borda Count method in that it is sorted by aggregate scores. However, DecideJoining focuses more on the historical performance and current status of the agent, while Borda Count focuses more on the preferences of voters.

If an agent has recently made positive contributions, these are given significant consideration, indicating a forgiving perspective even in the context of past errors or underperformance. This approach advocates for improvement and active involvement rather than judging agents solely based on historical behaviour. Such a strategy is instrumental in fostering the overall advancement and cohesive progress of the group.

These block diagram illustrates the allocation of varying weights to different attributes under different personalities in order to compute a resultant score.

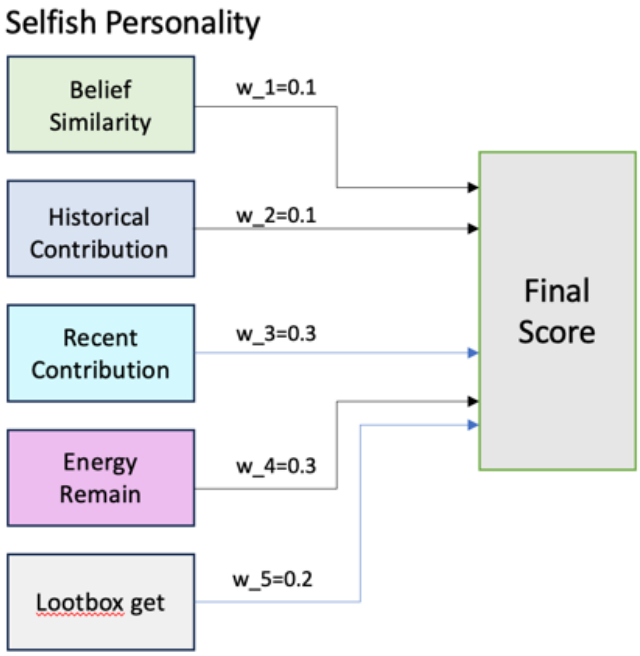


Figure 7.5: Different weight distribution of selfish personalities. (Allow other agents join)

In the case of selfish personality types, low weights are assigned to the similarity of beliefs and historical contributions. Selfish agents may display a diminished concern for the congruence of beliefs and long-term reputation based on past behaviours, favouring immediate self-interest unless it directly impacts the benefits. Given the propensity of selfish agents to prioritize actions that confer immediate advantages, greater weight is accorded to recent contributions and remaining energy, which is directly correlated with an agent's immediate survival and efficacy. Regarding the acquisition of Lootboxes, we assign medium to high weight, because selfish personalities often prioritise the potential benefits they might get from these Lootboxes.

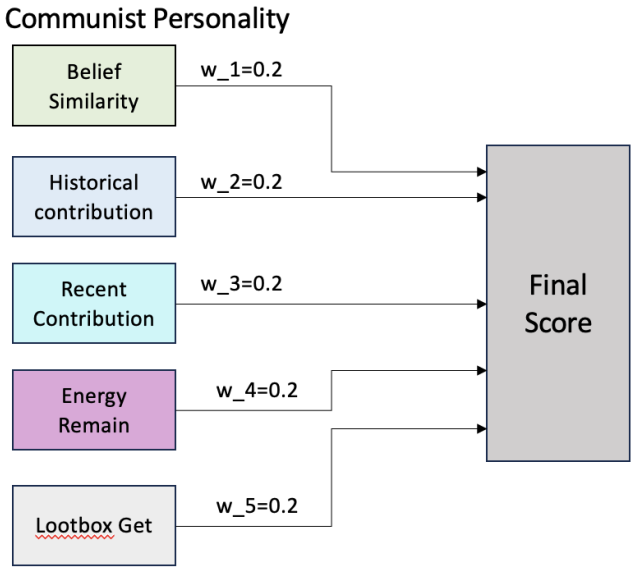


Figure 7.6: Different weight distribution of communist personalities. (Allow other agents join)

Belief similarity is assigned a moderate weight, as communist-type agents are inclined to share resources with others who have similar beliefs. Historical contributions carry a high weight, indicative of the agents' emphasis on long-term and equitable distribution of resources, making past contributions a significant factor in their decision-making. Recent contributions are also given moderate weight, as they offer insights into the agents' current behaviors and level of engagement. However, remaining energy is assigned a low weight, reflecting the communist type personality focus on overall balance and equity rather than the residual energy of individual agents. Regarding the loobox, we attribute it a low weight due to its outcome of collaborative effort.

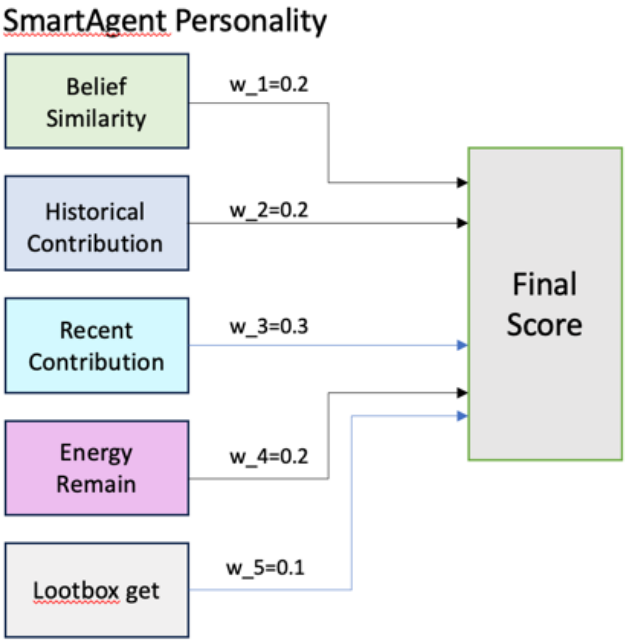


Figure 7.7: Different weight distribution of SmartAgent personalities. (Allow other agents join)

Smart agent personality is also known as team-oriented personality. Belief similarity and recent contribution is allocated high weights. Team-oriented personalities value congruence with the beliefs and goals of other agents in the team and the contributions of the agents to their immediate tasks on the bike. The historical contribution and energy remaining are given moderate weights, as we will consider the impact of teamwork history on future team activities and take into account individual residual energy to ensure that each agent can contribute effectively. Furthermore, we set Lootbox get to low weight as we value the overall performance of the team more than the gains of individual agents.

The 'DecideJoining' function implements the process by which an intelligent agent decides whether to allow other agents to join its bike. For each candidate agent (pendingAgents) that is pending to join, we will retrieve its reputation information from the reputation map and then rank the rankings. Based on the ranking results and the remaining capacity of the current bicycle, we decide whether to allow each agent to join.

7.3.3 Lootbox

Lootbox Selection Overall Strategy
In the process of selecting a Lootbox, the strategy was made from two aspects: the inherent benefits to the agent and the collective action problem. The strategy was based on score and ranking system. For each available Lootbox, the score would be computed according to 6 factors associated with the two aspects which would be further explained below. Subsequently, the agent would rank these Lootboxes, and ultimately, the Lootbox with the highest score would be selected.
Each factor under consideration was assigned varying levels of significance. To represent the importance of each factor in the final score, a specific weight was allocated to it. These factors were normalized, with values ranging from 0 to 1, to mitigate any disproportionate influences. Factors of greater importance had a greater effect on the score calculation. The assigned weights were adjustable, reflecting the distinct personalities of different agents. For a selfish agent, factors pertaining to its own benefit were given higher weights. In contrast, an agent with a communist personality prioritized factors related to collaboration with other agents. For a smart agent, all factors were taken into account to devise an optimal survival strategy.

Selfish Personality
A selfish personality meant the agent would only focus on self-satisfaction without considering others. The presumption of selfish agent was assuming all other agents would follow the rule, the agent can then choose the Lootbox with the highest energy contained and the matched color. This decision was made according to the free rider concept.
Communist Personality
An agent characterized by a communist personality tends to place a greater emphasis on collective interests,

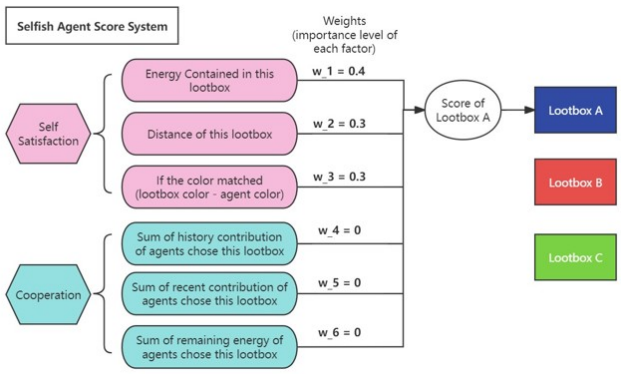


Figure 7.8: Selfish_Personality

striving to cooperate and maximize the benefits for all agents involved. The theory of fairness was implemented in this personality which tried to fairly allocate the energy.

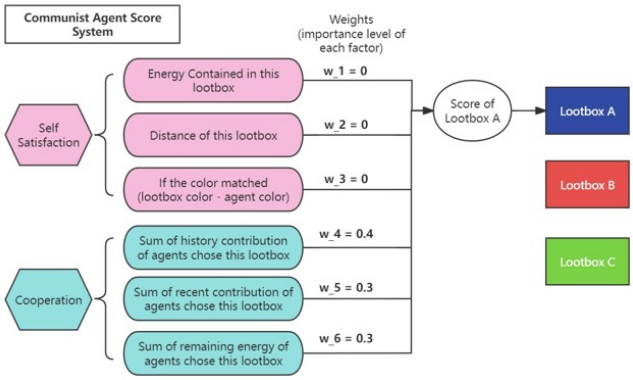


Figure 7.9: Communist_Personality

Smart Agent Personality
All factors should be considered reasonable. The two aspects were taken into account. A balance of self-satisfaction and cooperation should be found.

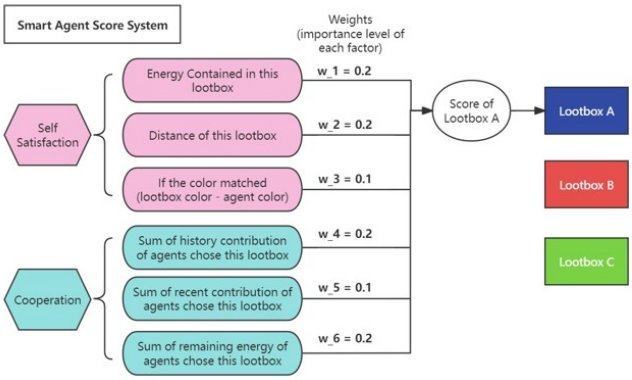


Figure 7.10: SmartAgent_Personality

Factors Considered in the Aspect of Self-Satisfaction
Considering the satisfaction of the agent itself, the energy contained in each Lootbox, the distance of each Lootbox and whether the color was matched with the agent's desired color were considered.

1. Energy contained in each Lootbox

The energy contained in each Lootbox was directly proportional to the agent's self-satisfaction, primarily because the primary criterion for survival in the game was the possession of adequate energy. The opportunity of

acquiring more energy would be higher with a Lootbox with more energy inside. Therefore, this feature – the energy contained in each Lootbox – would be assigned a positive weight when calculating the score of each box.

2. Distance of each Lootbox

The distance of each Lootbox should be inversely proportional to the score of the Lootbox. A nearer Lootbox would gain a higer score compared to a further Lootbox. The greater the distance was, the more energy the agent would need to reach the box. Consequently, agents would present a preference to choose a closer Lootbox which enabled them to conserve energy.

3. Color of each Lootbox

The consistency of the Lootbox's color with the agent's preferred color was assessed using a Boolean value (0 or 1). A match in color would enhance the agent's satisfaction due to the acquisition of bonus points. In contrast, a mismatch would not yield any bonus points. Therefore, a positive weight would be allocated to this feature.

Factors Considered in the Aspect of Cooperation
For each Lootbox, the information about which agents were intending to visit it, as gathered from the previous round, was available. Consequently, it was feasible to compute a cumulative reputation score for each Lootbox based on the reputation of the agents who had selected it.

In addressing the aspect of cooperation, it was essential to appraise other agents through a framework of opinion formation, which consisted of trustiness and forgiveness [44]. A reputation system was employed to contain all useful information related to other agents and quantify the evaluation of other agents. The risk of no effort can be decreased by considering these factors. The ideal Lootbox selection would enable our agent to collaborate with other agents possessing high reputation scores.

4. History Contribution of each agent

The measure of trustworthiness was based on past cooperative experiences with other agents within the game. Each agent's historical contributions reflected their performance and personalities. Our agent should be inclined to collaborate with those who have demonstrated a high level of historical contribution. These agents are deemed trustworthy, as they have consistently shown a tendency to exert effort in reaching the lootbox.

5. Recent Contribution of each agent

Forgiveness also played an important role in collaboration. The decision of each agent was dynamic during the game. An agent's past performance was not entirely indicative of its character or future actions. Therefore, the recent contribution needed to be considered. If an agent did not contribute much energy in past rounds; however, it worked hard in the recent two rounds. The smart agent can consider forgiving the agent at a reasonable level. Thus, a modest positive weight is assigned to this aspect of forgiveness in the score process.

6. Energy Remained of each agent

The remaining energy of each agent is indicative of their capability. An agent might be highly regarded in terms of opinion, but if they lack sufficient energy, their ability to contribute energy to collaborative was limited. Consequently, the remaining energy level is a significant factor in the evaluation process. Agents with lower energy reserved should be assigned a correspondingly lower score, reflecting their diminished capacity for assistance. Thus, the remaining energy is directly proportional to the overall assessment of an agent's value in the collaboration.

7.3.4 Voting for agents' proposals

Voting strategy
At the beginning of each round, every agent would deliver their preferences for their target Lootboxes. All agents need to vote for these proposals to pick one of them. The primary methodology employed by our agent during the voting process is the Borda Count method. It would assign points to each candidate based on their rank in each voter's preference list and then sums up these points to determine the overall ranking.
In the event of voting for elections for proposals, there are three kinds of personalities considered for our agent.

Smart Agent Personality
The smart agent employs a comprehensive approach to determine the assignment of scores to each agent, taking into account various factors derived from our reputation function, Key elements in this evaluation include the historical contributions and recent activities of each agent, as well as their remaining energy levels. Additionally, the assessment considers the resources contained in the targeted Lootbox, with a preference for those offering greater resources. The color of the Lootbox is another crucial consideration, as our agent exhibits a preference

for proposals featuring Lootboxes of the same color, thereby enhancing the likelihood of scoring. Moreover, the distance between the targeted Lootbox and our agent's bike significantly influences proposal determination, with a shorter distance being prioritized to enhance efficiency and minimize energy costs. The final proposals election would consider all these aspects mentioned with optimal weighted contribution to make a fair decision.

Communist Personality
The second is a communist personality for the agent, which focuses on collective well-being and the equitable distribution of resources among all agents. In this situation, self-benefit would not be essential anymore. More emphasis would be put on the collaboration. So, whether the color of the target Lootbox is what we want would not be considered in this agent state. Other agent's recent and historical contribution and their remaining energy would be a larger factor of the weighted contribution.

Selfish Personality
In a selfish personality, the agent's behavior is characterized by a focus on advancing its individual interests, aiming to optimize its personal utility or gains, and may not necessarily take into account the overall welfare or optimal result for the entire system. In this case, the reputation function is not necessarily to be considered. Our agent only considers finding the target Lootbox sharing the same color as ours, while simultaneously minimizing the distance to the Lootbox and maximizing the quantity of contained resources.
Different states of the agent would impact the final decision taken from the proposal pool resulting in different games' outcome. Agent would adjust the weighted contribution for every aspect in different states.

Algorithm for Democracy voting:

```
proposedLootBox <- the list of proposals we have
other_agent_id <- other agent ID
other_agent_proposed_lootbox_id <- other agent proposed Lootbox ID
loot := lootbox.GetTotalResources() / 4.0 <- resources contained in every target lootbox proposed
if lootbox.GetColour()==agent.GetColour()
is_color = 1.0 <- verify whether the proposals have the same color of lootbox as ours
other_agents_score <-reputation scores
distance <- Lootbox distance from the bike
normalized_distance<-normalized distance for Lootbox
scores := loot + is_color + normalized_distance + other_agents_score <-total scores
Different personalities would have different weights for these parameters
for other_agent_id in the range scores
scores[other_agent_id] <- get the rank of the proposals.
```

7.3.5 Pedal

Pedal Problem

In the MVP environment, each vehicle faces two main threats: firstly, individual vehicles may die due to energy depletion; secondly, there is a pouncer, Awdi, in the MVP world, which attacks vehicles with lower speeds. During the pedalling phase, each agent needs to contribute the appropriate amount of power depending on the direction decided to circumvent Awdi's pursuit and acquire the Lootbox faster, maximizing the collective benefit. However, for individuals, less effort can maximize individual benefits if the collaborator contributes more effort. Therefore, we want our smart agents to learn how to balance the conflicts between the individual and the collective so that they can make the most appropriate choices.

Strategy

We have developed a strategy designed to maximize our interests by developing agents who possess both refined egoistic qualities and are suitable followers. To achieve this goal, agents will gather information on historical decisions, and assess the similarity and leadership skills of other agents. And gain experience from the most successful agents and mimic their historical decisions, often referred to as the "lemming effect" [62]. In order to be able to quantify our evaluation of other agents, smart-agent selects the following parameters from Reputation to consider:
1. is-same-colour(considering the similarity in decision-making)

2. history-contributions (whether they were active in reaching the goal)

3. lootBoxGet (decision success rate)

4. recent contribution (determination)

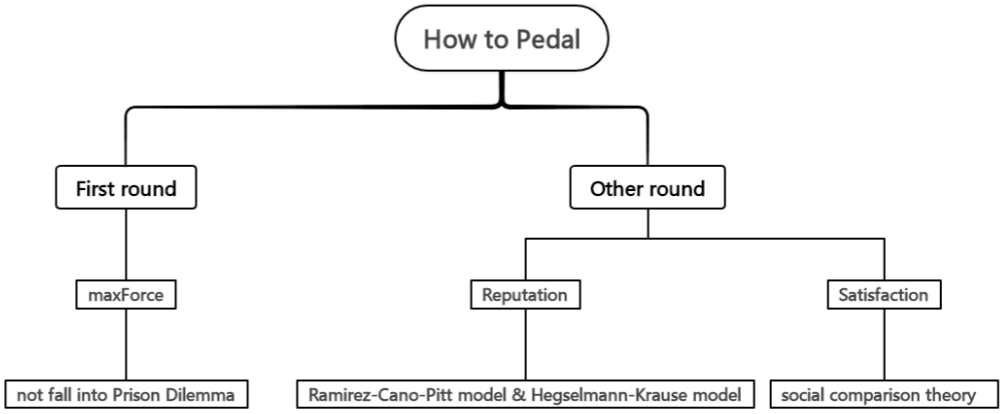


Figure 7.11: How to Pedal

Referring to the Hegselmann-Krause model [24], we plan to set the sum of the agents' scores to 1

$$\sum_{i=1}^{2} agentscore = 1$$

To achieve this, we normalize the relevant parameters of each agent to a single value, and each agent's score is divided by the sum of all the scores for its $agentscore$Next, the Smart-Agent selects the agent with the highest level of trust for the last round of decision-making as part of the Pedal Force. In addition, we will also consider the factor of Satisfaction. Satisfaction [20] is a measure of how similar the actual distribution is to our internal expectations and is used to reflect the degree to which citizens approve of the social distribution. This factor will also have an impact on the size of the Pedal Force.

This approach aims to enable agents to make decisions based on the behavior of other agents by simulating the trust-building mechanism of the Hegselmann-Krause model and taking into account the individual's acceptance of the distribution through satisfaction. Such a design is expected to find a suitable trade-off between balancing individual and collective interests, leading Smart-Agent to make more rational and socially acceptable decisions.

Implementation

For Pedal, we mainly use the data collected in the past to evaluate other agents in society and follow the person with the highest overall value in the team. Smart-Agent also fights against the unfair distribution of resources in society, the following pseudo-code expresses the logic of Smart-Agent in Pedal.

Algorithm for pedalling:

---

decide which Personality to use
Selfish: Maximising personal benefits
$score = 0.85 * SameColor + 0.05 * historyContribution + 0.05 * lootBoxGet + 0.05 * recentContribution$
Communist: Maximising the benefits for the team
$score = 0.015 * SameColor + 0.5 * historyContribution + 0.34 * lootBoxGet + 0.15 * recentContribution$
Smartagent: Team and individual balance
$score = 0.3 * SameColor + 0.2 * historyContribution + 0.3 * lootBoxGet + 0.2 * recentContribution$
Initialize:
In the first round, We want to make a good impression on the other agent.
Pedal=Max

Then:
Get historical performance for all agents
Calculate the Score of each agent
Normalize and sort the data
Choose the highest agent as leader and take his past decisions into consideration
Obtaining social satisfaction (Satisfaction is a value that has been quantified$\in$[0,1]
Calculate the final pedal force

if force > Max:
Return Max
else
Return Pedal force

7.3.6 Energy Allocation

Upon acquisition of a Lootbox, the allocation of its energy content necessitates a nuanced approach, wherein our agent manifests distinct behavioral traits across three specific characteristics: selfish, fairness, and smart agent. This multifaceted behavior underscores the agent's adaptability and responsiveness in orchestrating the equitable distribution of energy among the members situated on the bike. The delineation of these characteristics enables our agent to exhibit nuanced decision-making, aligning with the dynamic contextual requirements that may arise during the energy distribution process. This sophisticated approach ensures that the distribution strategy is intricately tailored to varying circumstances, fostering an optimal and harmonious energy-sharing dynamic within the cooperative framework of the agent ensemble.
Adhering to principles of distributive justice, fairness stands out as a paramount criterion to uphold which is described in J. Pitt's paper as follows: [30]

- *Proportional: Each agent is allocated a fair share of resources, specifically* $1/nth$ *of the total allocation, where* n *is the number of agents.*
- *Envy free: Each agent is content with their assigned resources and does not envy others.*
- *Equitable: Every agent derives the same level of utility from their allocated resources. The distribution is designed to ensure an equal level of satisfaction among all agents.*
- *Efficient: The allocation aims to maximize overall utility by ensuring that resources are directed to those who need them the most or can make the most effective use of then. The goal is to achieve the greatest benefit for the greatest number of agents.*
- *Cost-effective: The computational process for determining the resource distribution does not consume a disproportionate amount of resources. This is particularly crucial in systems where the computation of the allocation must be "paid for" using the same resources that are being allocated.*
- *Timely: The computation of the allocation terminates within a reasonable timeframe, ensuring that the distribution process is efficient and does not delay the utilization of resources.*

Selfish Personality

The agent always gets all of the gained energy when the bike gets a Lootbox.

Communist Personality

The agent consistently advocates for the equitable distribution of energy among all members on the bike, grounded in the principles of both equality and need. This approach reflects a conscientious consideration for the well-being of those least advantaged within the societal construct, aligning with a commitment to fairness and distributive justice.

Smart Agent Personality

The agent adheres to a nuanced strategy for energy distribution, referencing the predefined reference map when its own energy surpasses a specific threshold. Conversely, when its energy falls below this threshold, the agent exclusively proposes allocations based on need, specifically, the remaining energy. This dual-tiered approach is underpinned by a conscientious regard for the well-being of those at a relative disadvantage. Additionally, it aligns with principles of equity and desert, advocating that an individual should receive an allocation proportionate to their contributions to the societal framework. This methodology ensures a judicious and context-sensitive allocation strategy that accommodates both overarching societal considerations and individual contributions.

7.3.7 Message

Basically, our agents will honestly share our decisions and relevant information which not accessible from the environment, to other agents in the same Mega-Bike. Our agents will inform fellows in same Mega-Bike about desired Lootbox and the agent with the lowest reputation on board. If an agent's reputation falls below a threshold (half of the average level), it's suggested to consider expelling them from the Mega-Bike.

Our agent will also inform other fellows about the pedalling decision. However this is the only message which is not 100% honest. In this case, our agent will exaggerate the force we devoted in according to how fairness we find the recent allocation on the Mega-Bike is. This is similar to what we take into consideration when we decide how much to pedal. The function to generate the force we mentioned in message is shown bellow

$$Pedal_{message} = \frac{Pedal_{truth}}{satisfaction} \tag{7.5}$$

Our agents only pay attention to messages about target Lootboxes and requests to kick off someone from current Mega-Bike. If these align with our calculations, we will increase opinion similarity rating for those agents which means considering their needs and proposals more in future decisions. Messages containing too general (like governance related) or too specific (like detailed allocation maps) information are not processed.

7.4 Experiment and Analysis

We have established **three distinct simulation scenarios** to assess the viability of specific agents. In the first scenario, our team's agent operates in isolation; in the second, our agent coexists with a baseline agent; and in the third, all agents compete in a shared environment.
Within these scenarios, we introduced agents with **three different personalities (smart, selfish, communist)** and evaluated their performance across **three key metrics: lifetime, energy average and points average.**

7.4.1 Experiments Result

The results were the average values from 10 rounds experiments.

	Lifetime	**EnergyAverage**	**PointsAverage**
smart	74.284	0.722	5.4
selfish	71.546	0.714	5.072
communist	71.7	0.708	5.202

Table 7.2: Performance of agents with varying personalities in all-agents competitive environment.

7.4.2 Performance Comparison

Comparsion and Analysis of Points achieving
In the context of a 10-round game involving seven other agents, it was evident that our smart agent would achieve the highest score, approximately 5.4, surpassing both the communist and selfish agents. This observation underscored the advantageous outcomes associated with the smart personality trait, as it not only considered individual benefits but also prioritized team cooperation. Conversely, the communist agent, while demonstrating a focus on others' performance, tended to yield slightly lower benefits for the individual. Ideally the selfish personality, primarily concerned with self-benefit, would secure the highest scores among the three personality types. However, in the multiplayer virtual environment (MVP), challenges might arise in terms of interpersonal connections and trust, potentially leading to the exclusion of a consistently self-centered agent from the team. In this case, during the scenario, the agent with selfish personality gained the lowest scores compared to other personalities.

Comparison and Analysis of Lifetime
The minimal average life time recorded across ten experimental rounds was 71.546 with selfish personality. In cooperative problem-solving scenarios, a selfish personality typically results in diminished esteem amongst peer agents. A consequential low reputation hinders cooperative engagements, as other participants are reticent to ally with individuals perceived as potential free-riders. Furthermore, these agents are less inclined to share energy, with those exhibiting selfish behavior, especially given the latter's tendency not to contribute equitably during collective tasks. Additionally, during decision-making processes involving voting, proposals tendered by

agents with a selfish personality are often met with disfavor. Therefore, a selfish personality had the shortest lifespan.
The maximum average lifespan observed was 74.284, achieved in scenarios where agents exhibited a 'smart agent' personality. This personality type was characterized by its comprehensive evaluation of other agents, coupled with a keen consideration of its own benefits. The smart agent employed a multitude of strategies to facilitate reasonable decision-making. For example, it assessed other agents using a social framework, which involves analyzing their behaviors and energy levels from past rounds. Additionally, it calculates the reputation of each agent to gauge the risk of trust which can decide the reliability and potential cooperation level of other agents. Thus, the comprehensive consideration in the smart agent personality can result in a longer average lifespan within the experimental framework.
With a communist personality, the lifespan was longer than that observed in selfish personalities, yet significantly shorter than in smart agent personalities. The reason is that the communist personality tends to gain a high opinion among other agents, leading to increased willingness for cooperation. However, in the presence of more selfish agents, the communist personality may end up giving more and receiving less. Consequently, in environments lacking other communist agents, the lifespan of a communist personality agent tends to be shorter.

Comparison and Analysis of Energy level
In the scenario that involved 8 team agents for 10-round, there were distinct differences. Our agents, when adopting a communist personality, tended to prioritize the team's benefit over individual gain, often at the expense of their own energy levels. Those with a selfish personality risked receiving less energy during allocation as a consequence of their actions, resulting in only a marginally higher energy level than the communist agents, despite their free-rider approach. Meanwhile, our smart agents not only emulated others but also consistently projected a positive attitude across the team. This led to the conclusion that our smart personality demonstrated superior performance in interactions with other agents.

7.5 Conclusion

The smart agent demonstrates superior performance when compared to other personality types in terms of lifespan, energy utilization, and overall scores. This is attributed to its dual consideration of self-benefit and cooperative tendencies. Conversely, the selfish personality exhibits the poorest outcomes in both lifespan and score acquisition due to its negative reputation, limited social support, and challenges in collaborative efforts. The communist agent, focusing on teamwork and energy distribution, performs inadequately in the energy aspect, resulting in the least energy retention. This scenario highlights that a personality that balances self-interest with a collaborative approach achieves optimal performance.

7.6 Future Work

The future work of our project will introduce dynamic personality-switching for the intelligent agent, aiming to enhance survival and adaptability of our agents in the game's fluctuating environment. This innovative approach will enable the agent to adjust its behavior in response to specific thresholds such as energy levels and the perceived effort of other groups. Possible suggested improvements for future work are as follows:

Personality-Switching Mechanism:

Develop a mechanism that allows the agent to switch between team-oriented, selfish, and communist personalities based on real-time game metrics. This will involve setting critical thresholds that, when triggered, will prompt a change in the agent's strategy personalities.

Adaptive Survival Strategy:
Implement an adaptive survival strategy that prioritizes the agent's longevity by evaluating the current state of the game. For instance, if energy levels of our agents are critically low or if there is evidence of other groups freeriding, the agent will switch to a selfish personality to preserve its resources.Furthermore, if the system results after every round end detected that other groups are nearing depletion, then we can be activated the communist personality. By sharing resources and cooperating to achive a longer survival duration for all the agents, including ourself, as the agent aims to ensure collective endurance. After each game round, the agent will assess the performance of all groups, including its own, to determine the optimal personality mode for the next round.

Enhanced Learning Algorithms:
Explore the integration of machine learning algorithms that can learn the optimal timing and conditions for

personality switching from historical game data, enhancing the agent's decision-making process over successive game iterations.

7.7 Project Management

The project organisation for Team 3 consists of four main phases. In each phase, each team member undertook a different task, and the table below shows what each team member did.

Name	Research	Design MVP	Post MVP	Complete Agent
Xinlong Tan	Research	Environment design	Coding/Environment/speaker	Simulation/Debug
Ruiqi Shen	Research	Environment design	Coding/Debug/governance/leadership	Simulation/Debug
Yuyao He	Research	Internal design	loot box/voting/Pedal	Final report/Graphics
Xuanyu Lai	Research	Internal design	Pedal/Satisfaction/allocation	Final report/Tuning
Zhiyu Ma	Research	Infrastructure Rep	Allocation/Personality/Presentation	Final report/Tuning
Tongyu Ding	Research	Infrastructure Rep	Voting/reputation/loot box	Final report/Graphics
Zekai Yushi	Research	Environment design	Presentation/reputation/Voting	Final report/Simulation
Zhoujing Yuan	Research	Internal design	Reputation/leadership/Personality	Final report/Tuning

Figure 7.12: Project Team Roles and Responsibilities

Phase 1 Background research

As the project started, based on the requirements of the project, each team member was assigned a specific role. Focusing on background research and preliminary design. We did this by organizing the theories from the lectures and looking for papers related to the theories. At this stage, we gained an initial understanding of the project framework. At the same time, regular team meetings facilitated the sharing of research insights and ensured alignment of individual contributions with the project's overall direction.

Phase 2 MVP design

During this stage, the focus shifted to the development of a prototype. We collaborated with other groups to outline the MVP's structure, including the sequence of events and the design of the predator Awdi. The interaction of the agent with the environment and other agents is the key point. For our group's agent, we prioritized functionalities like voting, pedalling, turning and speaker roles. To manage this complex task, we divided the team into three sub-teams: Environment Design, Internal Design, and Infrastructure Design. Each sub-team took charge of implementing the designated functionalities, ensuring an efficient distribution of the workload.

Phase 3 Post MVP design

With the core objectives of our intelligent agent established, the project entered a crucial phase of management and strategy formulation. Our approach was methodically structured to ensure each component of the agent's design was satisfied. It was required to meet the functional and performance criteria we had set. Upon finalizing the MVP, we embarked on implementing and testing our agent's strategic functions. Each team member is assigned a specific development task for each feature function. To ensure consistency and collaboration among team members, we compiled a 'common state' checklist enumerating shared variables and parameters within each function.

Each sub-team was tasked with specific responsibilities, not only to ensure their modules were aligned with the overall strategy of the intelligent agent but also to facilitate seamless integration within the Go language environment. This collaborative effort guaranteed that the main implementation of the agent was accomplished in time for the internal agent deadline.

Phase 4 Final MVP design

Following the initial implementation of our agent's strategy, we entered the final phase of the project—optimizing the strategy and simulating and tuning our agent with other agents on MVP. Key areas of focus for the team during this stage included:

Simulation Testing: Conducting multiple rounds of simulations to test the agent strategy with various parameters, documenting behaviors and outcomes under different strategic parameter settings to assess and refine the agent's performance.

Parameter Tuning: Identifying the optimal settings for the strategy's default parameters that create the most advantageous behavior in interactions with other teams.

Reporting: Detailing each function of the agent's strategy and experiment simulation results, collaboratively writing the final team and collective report.

Debugging: We established a rigorous debugging protocol involving systematic testing and code review. Our team participated in collaborative debugging sessions where we could collectively review and troubleshoot sections of the code and ensure each decision-making function underwent a thorough logic verification process. By stepping through each decision point, we ensured that the agent's logic was sound and that it would make the best possible choices given its current state and environmental conditions.

Graphics: We designed a series of engaging and informative graphics that demonstrated the agent's capabilities and parameters, ensuring that data is visualized.

Each team member contributed to a section of the final report, ensuring a considerate presentation of the agent's capabilities and strategic functions. The phase was the deployment of the agent into a final simulation environment, getting the multiple type test outcomes that the performance of our agent within the MVP world.

Chapter 8: Team 4

8.1 Introduction

The main goal of our agent in SOMAS World is to survive as long as possible in an ever changing environment.Our strategy has been designed to act on different systems of governance, communication and social adaptability. The foundations of our agent was built on three dynamic features namely an honesty matrix, reputation matrix and structure. Both the honesty and reputation matrices gave our agent social awareness, while the structure (**brain**) initiates actions and decisions that aligned with our agents' mission. As more theories were added and functions became more complex, the ability to predict our agents result became more of a mystery. By using this all-encompassing strategy, it not only adjusts to its environment but also actively makes decisions based on its learning . Our agent slowly developed a personality mimicking that of a "human mind" ... revealing the hidden beauty of multi agent systems.

8.2 Environment & Ideation

Our first step, as a group was to learn and experiment with the environment using the provided BaseBiker and default functions of the server. By learning the environment and the steps that our agent was required to take, we were able to formulate an approach that covered all the fundamental concepts to successfully complete the game. We asked ourselves questions that correlated with the fundamentals of "Game Theory"[55], What factors should we consider when we make a decision? Should our strategy change as the game evolves? What parameters should we consider when formulating an opinion or taking action? etc.

Through the initial testing, we were able to discover reoccurring patterns and important parameters in the game that we should keep in mind in order to design our agent.

The tests were implemented on key areas such as:

- Different Voting Strategies and Methods
- Conservative Approach, emphasis on safety
- Radical Approach, emphasis on return

Strategy Ideation:

- Agents were extremely dependent on each other regardless of their "social role" on the Mega-Bike. With this in mind, we decided to utilise our response and interactions with other agents in the environment. Furthermore our agentś chance of survival increased by emphasizing the importance of reputation, tying in with our main goal.
- Agent would follow a "Risk vs Return Analysis", whereby safety is favored rather than taking unnecessary risks that have higher returns. Strategies that risk our survival in the game are associated with higher cost.
- Agent will be fair in both its decisions and actions in order to build a "Reputation" and "Trust" amongst other agents.

Overall the strategy is divided into three components namely honesty, reputation and action (structure). These components will retrieve information from the environment, interpret it and implement an action that correlates with our agentś goal.

8.3 Background

The concept that forms the foundation for our agent is the "Ramirez-Cano-Pitt (RCP) model" [61]. The reputation matrix and honesty matrix which helps in the decision making of our agent form a framework

similar to the RCP model. These matrices are dynamically updated throughout the game. The RCP model is an extension of Hegelsmann-Krause model [22]. It[61] consists of three key parameters: opinion, affinity and confidence. The opinion is a dynamic parameter which changes based on the affinity of an agent with other agents. The confidence parameter acts as the weights for the opinions. The opinion in our case is the reputation value of each agent set by our agent. The affinity is the agent's interaction with other agents based on which the agent learns the behaviour of the other agents with which it interacts (i.e possibly agents in the same Mega-Bike). Based on the learning the reputation value is updated. The confidence is the honesty value of each agent obtained from the honesty matrix (section below).This confidence value helps to decide how much to trust a particular agent.

8.4 Reputation Matrix

8.4.1 Design

The reputation matrix is created to compute the reputation value of each agent. Our own agent is given the highest reputation as we consider our agent to be fair. The two factors considered to measure the reputation of an agent is the Energy Reputation and Leaving Reputation.

Energy Reputation

It draws a direct correlation between an agent's energy consumption and its perceived level of selflessness. The premise is straightforward: the more energy an agent expends, the greater the perceived effort and selflessness. This is encapsulated in the notion that an agent expending the majority of its energy is indicative of a higher degree of commitment and dedication. In contrast, an agent that conserves energy may be seen as expending less effort, reflecting a potentially lower level of selflessness. It also checks not just the exact energy consumption but also how much of their overall energy have they expanded since different agents will have different energy levels. Equation 8.1 shows how Energy Reputation is calculated.

Let *myConsumption* and *myEnergyLevel* be the consumption and energy level of the agent forming an opinion (our agent), and let *consumption* and *energyLevel* be the corresponding attributes for the agent being judged. The energy reputation E_{rep} can be defined as:

$$R_{Energy} = \left(\frac{myConsumption}{myEnergyLevel + 0.001}\right) - \left(\frac{consumption}{energyLevel + 0.001}\right) \tag{8.1}$$

Leaving Reputation

This is a vital indicator of the agent's contribution to the Mega-Bike. The agents who were kicked out were most probably greedy with the resources or did not make any significant contribution. They ultimately let their group down even if they left voluntarily. Thus, when an agent agent leaves a bike , their reputation value is decreased.

The Reputation concept not only encourages cooperative behaviours but also fosters an environment where agents are motivated to contribute positively to the group's objectives. It aligns with the overarching goal of building a community of agents that actively work together, reinforcing the importance of collaboration and shared efforts within the dynamic landscape of SOMAS World. Equations 8.2 and 8.3 show how Leaving Reputation is calculated.

Let $R_{BikeShift}$ be a variable that denotes the reputation shift due to being part of a Mega-Bike. It can be defined conditionally as follows:

$$R_{BikeShift} = \begin{cases} 0.2 & \text{if the agent left a bike,} \\ 1.0 & \text{otherwise.} \end{cases} \tag{8.2}$$

$$R = R_{Energy} * R_{BikeShift} \tag{8.3}$$

8.4.2 Implementation

Algorithm 1: CalculateReputation(agent)

```
Data: agent
Result: None
megaBikes ← agent.GetGameState().GetMegaBikes();
decayFactor ← 0.1;
totalReputationSum ← 0;
foreach bike in megaBikes do
    fellowBikers ← bike.GetAgents();
    foreach otherAgent in fellowBikers do
        // Set initial reputation to 1.0 for all agents, except ourselves (0.0)
        if otherAgent is agent then
            agent.reputation[otherAgent.GetID()] ← 0.0;
        else
            consumption ← CalculateConsumption(otherAgent);
            myConsumption ← CalculateMyConsumption(agent);
            // Calculate energy reputation
            energyReputation ← (consumption/(otherAgent.GetEnergyLevel() + 0.001)) −
              (myConsumption/(agent.GetEnergyLevel() + 0.001));
            // Determine reputation shift based on bike status
            reputationBikeShift ← DetermineReputationBikeShift(agent, bike);
            // Calculate overall reputation
            overallReputation ← energyReputation × reputationBikeShift;
            // Apply sigmoid function to scale reputation between 0 and 1
            overallReputation ← ApplySigmoid(overallReputation);
            // Finalize and update reputation with decay factor-to consider previous
               reputation value
            finalReputation ← (1 − decayFactor) × agent.reputation[otherAgent.GetID()] +
              decayFactor × overallReputation;
            agent.reputation[otherAgent.GetID()] ← finalReputation;
            totalReputationSum ← totalReputationSum + finalReputation;
foreach bike in megaBikes do
    fellowBikers ← bike.GetAgents();
    foreach otherAgent in fellowBikers do
        // Normalize reputation values between 0 and 1 within each bike
        agent.reputation[otherAgent.GetID()] ←
          agent.reputation[otherAgent.GetID()]/totalReputationSum;
```

The algorithm 1 calculates the reputation of all agents considering the both the factors of the Reputation matrix along with the past reputation value. This is used to make all major decisions. The reputation value is recalculated in every iteration to improve its accuracy and so our agent can learn while playing.

8.4.3 Output

The reputation matrix has been created for all agents with 2 columns. One column will have the agent's unique ID and the second column will have their reputation value. It will be a number between 0 and 1.

8.5 Honesty matrix

8.5.1 Design

Expanding upon the "Trust" concept , our agent incorporates a dynamic honesty matrix to utilise the fluid nature of trust in the SOMAS World landscape.

To comprehensively evaluate trust, our design considers the dual dimensions of uncertainty elucidated by Weisberg (2014) [75]: doubt and ambiguity. Doubt is quantified by each agent's honesty matrix value, providing a numerical representation of the level of uncertainty associated with their communication. Simultaneously, ambiguity is a qualitative measure that involves the agent understanding the current situation and determining the severity of errors committed by other agents. In cases of trust violation, the agent subtracts points from the honesty matrix proportionate to the gravity of the mistake. The honesty value of an agent decides how confident we are about another agent's opinion based on the RCP model [61].

Additionally, our model incorporates a forward-looking element by introducing forgiveness. Points are added to the honesty matrix if another agent demonstrates improved performance over multiple iterations. This forgiveness mechanism acknowledges the potential for positive change in an agent's behavior and rewards such improvement in subsequent rounds.

Applying Condorcet's Jury Theorem to Trust Evaluation [70], In the context of trust evaluation, Condorcet's Jury Theorem can be analogously applied to the decision-making process of the agents. Let p be the probability of an agent making the correct decision in evaluating the trustworthiness of a communication, n be the total number of agents involved, and P be the probability that the majority decision among the agents is correct. The theorem suggests that as the number of agents n increases, and if each agent has a probability p greater than 0.5 of making a correct trust assessment, then the probability P that the majority will make a correct decision also increases.

The formula for P in our trust evaluation context is given by:

$$P(n,p) = \sum_{k=\lceil \frac{n}{2} \rceil}^{n} \binom{n}{k} p^k (1-p)^{n-k} \tag{8.4}$$

where $\binom{n}{k}$ is the binomial coefficient, representing the number of ways to choose k correct trust assessments out of n independent evaluations, and $\lceil \frac{n}{2} \rceil$ denotes the smallest integer greater than or equal to $\frac{n}{2}$. Equation 8.4 represents the idea that we will have more accurate assessment of the honesty of an agent after multiple iterations which is better than to to hold on to preconceived notions about other agents.

In summary, this comprehensive framework not only captures the evolving nature of trust through a dynamic honesty matrix but also considers both quantitative and qualitative aspects of uncertainty. The incorporation of forgiveness adds a layer of adaptability and encourages a collaborative environment where agents have the opportunity to rebuild trust through sustained positive behavior over time.

Messaging

We are using all 5 types of messages available for communication. We are always honest in our communication and do not lie based on another agent's reputation or honesty value as we want to be fair and just and treat other agents the same way we want to be treated.

8.5.2 Implementation

It assumes all agents are honest initially and decreases it, as it gets to know them better. Agents are penalized for dishonesty, that is when they communicate one thing and do something else instead.

Given in algorithm 2 is an example of the pseudo code for the honesty matrix. The severity of the situation is assessed using the values in the table 8.1. The values in the table were set based on our observations after multiple iterations. Based on the message received or each iteration in the game, honesty matrix value for each agent is updated using the given table 8.1 and algorithms 2 ,3,4.

Situations	Value
`if other agent wants to Kick us Out Value`	0.2
`reputation Threshold`	0.65
`if Reputation Is Low`	0.1
`like LootBox`	0.2
`dislike LootBox`	0.05
`same Color Agent in Democracy`	0.1
`same Color Agent in Leadership`	0.05
`same Color Agent in Dictatorship`	0.05
`different Color agent in Democracy`	0.1
`different Color agent in Leadership`	0.05
`different Color agent in Dictatorship`	0.05
`if Forces are TooLow`	0.4
`if Ruler is SameColor`	0.1
`if Ruler is Different Color`	0.1
`if We Are Ruler`	0.2
`if they KickOut Us`	0.5

Table 8.1: Situation Values

Algorithm 2: Calculate Honesty Matrix

Data: None
Result: HonestyMatrix (2D array)
Initialize HonestyMatrix as a 2D array with dimensions (numAgents, numAgents);
Set all honestyValues of each agent in HonestyMatrix to 1.0;

Algorithm 3: Increase Honesty Value

Data: agent ID , amount (real number)
Result: agentID.HonestyValue (real number)
Set UpdatedHonestyValue = agentID.honestyValue + amount;
if *UpdatedHonestyValue > 1.0* **then**
  Set UpdatedHonestyValue to 1.0;
else
Set agentID.honestyValue =UpdatedHonestyValue

Algorithm 4: Decrease Honesty

Data: agentID.honestyValue (real number)
Result: UpdatedHonestyValue (real number)
Set UpdatedHonestyValue = honestyValue - amount;
if *UpdatedHonestyValue < 0* **then**
  Set UpdatedHonestyValue to 0;
Set agentID.honestyValue =UpdatedHonestyValue

Note: The amount for the IncreaseHonesty and DecreaseHonesty function is chosen depending on the situation and values are taken from the Table 8.1

Algorithm 5 is an example of how the honesty value is updated for an agent based on communication received about governance. The data s̈enderColorr̈efers to the color of the agent who sent the message about governance. It uses algorithms 3 and 4 to compute the honesty value for an agent.

Algorithm 5: Receive Governance Message

```
Data: ourAgentColor, senderColor, governanceType
Result: UpdatedHonestyValue
if ourAgentColor is the same as senderColor then
    switch governanceType do
        case democracy do
            IncreaseHonesty by sameColor agent in Democracy;
        case leadership do
            IncreaseHonesty by same Color Agent in Leadership;
        case dictatorship do
            DecreaseHonesty by same Color agent in Dictatorship;
else
    switch governanceType do
        case democracy do
            IncreaseHonesty by different Color Agent in Democracy;
        case leadership do
            DecreaseHonesty by different Color agent in Leadership;
        case dictatorship do
            DecreaseHonesty by different Color agent in Dictatorship;
```

So if an agent is communicating with the agent is the same color – honesty is increased since we trust them more as they will have shared goals. But an agent of a different color can only be trusted if they vote for democracy since that ensures that we get an equal vote as them. There are many other situations where this value is updated like for all the 5 types of communication (like joining the bike, reputation message, kickout, and Lootbox messages as well).

8.5.3 Output

The honesty matrix is a value between 0 and 1 and this value is updated throughout the game. This value is updated every iteration – thus, over time it decides the true nature of the other agents and should be able to understand them better. In case an agent's actions improve and help our agent, that agent's honesty score will be increased as well.

8.6 Structure

8.6.1 Design

The structure could be considered the **brain** of our agent as it both receives information from the environment, interprets the output from the honesty and reputation matrices, and initiates the best action that our agent should perform to improve its state. The structure is implemented through four components- Allocation, Decision, Direction and Opinion, as shown in the diagram 8.1.

The figure 8.1 shows the exact path (input to output) that a signal would follow to trigger a response from the agent:

- Optimal Action (Block 1):

  Block 1 will calculate our agents "optimal action" excluding any social external factors. This input *("best move")* will only include internal factors such as the nearest Lootbox, our position relative to the Awdi, and our energy levels. This could be described as the optimal path that our agent should follow if the environment was static and if other agents remained neutral in their decisions and actions.

- Honesty Matrix (Block 2):

  Block 2 will include the final output from the honesty matrix (Communication and action correlation), the matrix will be indexed to retrieve information about fellow bikers and is incorporated into functions that may require any honesty information. This parameter is integrated as a dynamic variable and is weighted corresponding to our approach (weights can be changed depending on a relaxed or aggressive strategy).

- Reputation Matrix (Block 3):

  Block 3 will include the final output from the reputation matrix (Energy Levels, Decisions and Voting actions), the matrix will be indexed to retrieve information about fellow bikers and is incorporated into functions that may require any reputation information. This parameter is integrated as a dynamic variable and is weighted corresponding to our approach (weights can be changed depending on a relaxed or aggressive strategy).

- Structure Function (Block 4):

  Block 4 represents the general structure that each function would follow. The function inputs our optimal path which only includes factors that our agent can control and combines it with social external factors that may impact our actions and decisions. The optimal path is then refined to obtain the best outcome.

- Refined Action (Block 5):

  Block 5 represents the final output for the agent. The Refined Action will contain both external and internal factors for our agent to base its decisions on. Block 5 will trigger the best action and decision (when required) incorporating our agentś current status and social elements of honesty and reputation of other agents.

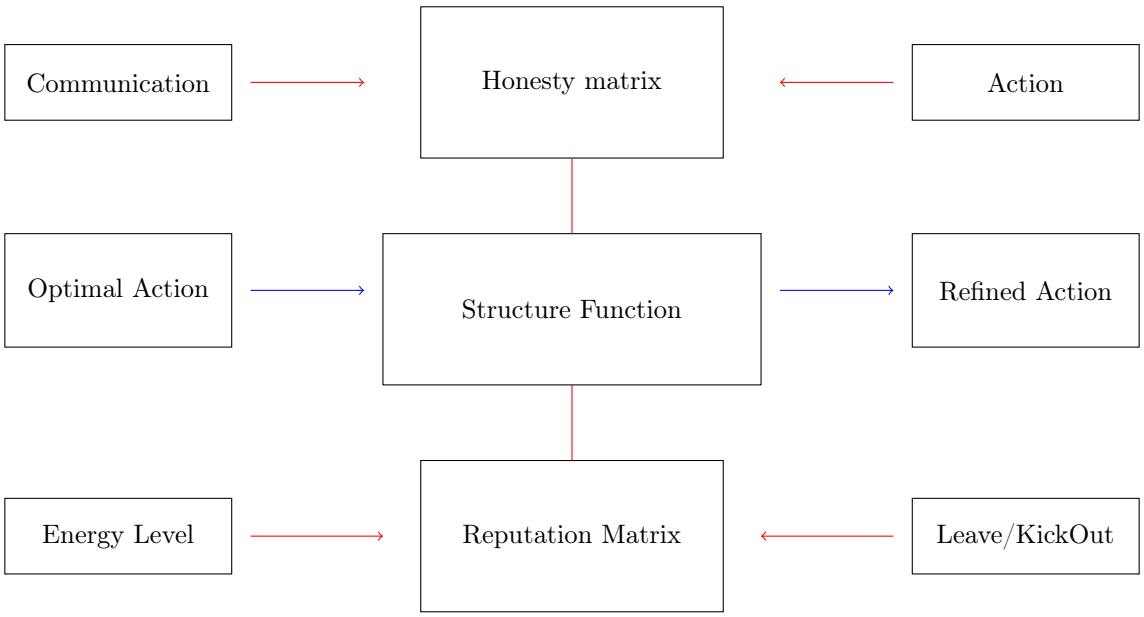


Figure 8.1: Block Diagram showing the design of our structure

8.6.2 Implementation

Key Functions:

Decision

The majority of the decisions that our agent takes are based on the reputation and the honesty values. There are certain exceptional situations where we deploy a different strategy. While considering to change bikes we would prefer to shift to a bike which has a considerably large number of agents with the same color as us. Algorithm 6 represents the scenario in which we assign larger weights to a bike with agents of our color than the other bikes. That is, we prefer to be in a majoritarian tyranny[55] based on the color. This would facilitate our agent to maximize the points obtained.

Algorithm 6: Update `bike priority` based on biker's color

```
if biker.GetColour() == agent.GetColour() then
    bikeWeight += 1.8;
else
    bikeWeight += 1;
```

Another situation where we utilize a different strategy is presented in Algorithm 7. It is a part of the function which decides weights, when our agent is the leader. It assigns equal weights to all the other agents' opinions or votes. That is, our agent tries to create a situation similar to that of a democracy. This facilitates increase of our agentś reputation which indirectly increases our chance of surviving longer and maximizing the points.

Algorithm 7: Update `weights` based on `fellow.GetID()`

```
if fellow.GetID() ≠ uuid.Nil then
    weights[fellow.GetID()] = 1.0;
else
    weights[fellow.GetID()] = 0.0
```

Allocation

Our agent follows distributive justice[47] to allocate resources to other agents. The distributive justice is rewarding each agent based on its performance. Here the agent considers reputation, honesty, energy spent and the rate of change of energy. Each parameter is weighted by the default weights considered by the agent. Two exceptions are present, like when it comes to allocating for our own agent, it demands a higher energy than the other agents. Another exception is when the Lootbox is of the same color as our agent, then it tries to allocate even more energy. This is done with the motive of increasing the life expectancy of our agent. Algorithm 8 explains the aforementioned allocation strategy.

Algorithm 8: `Resource allocation`

```
distribution[fellow.GetID()] = float64((reputationWeight × reputationRank[fellowID])+
        (honestyWeight × honestyRank[fellowID]) + (energySpentWeight × energySpent)+
                                    (energyLevelWeight × fellow.GetEnergyLevel()))

if fellowID == agent.GetID() then
    distribution[fellow.GetID()] = float64((reputationWeight × reputationRank[fellowID])+
        (honestyWeight × honestyRank[fellowID]) + (energySpentWeight × energySpent × 1.5)+
                                    (energyLevelWeight × fellow.GetEnergyLevel()))

    if agent.lootBoxColour == agent.GetColour() then
        distribution[fellow.GetID()] = float64((reputationWeight × reputationRank[fellowID])+
            (honestyWeight × honestyRank[fellowID]) + (energySpentWeight × energySpent × 1.5)+
                                    (energyLevelWeight × fellow.GetEnergyLevel() × 1.5))
```

Direction

The direction for which our agent votes varies based on numerous conditions. The highest priority is to escape from the Awdi and grab a Lootbox with our own color. If our agent encounters an Awdi, it tries to move in the opposite direction. The other possibility our agent considers is that the Lootbox's color matches another agent's color. An important criterion that our agent considers to vote for the nearest Lootbox or a distant Lootbox is the energy level. Algorithm 9 decides on the turning forces which gives the direction.

Algorithm 9: Biker direction

```
if distanceFromAwdi < awdiDistanceThreshold then
    deltaX ← awdiPos.X − currLocation.X
    deltaY ← awdiPos.Y − currLocation.Y
    angle ← atan2(deltaY, deltaX)
    normalisedAngle ← angle/π
    if normalisedAngle < 0.0 then
        flipAngle ← normalisedAngle + 1.0
    if normalisedAngle > 0.0 then
        flipAngle ← normalisedAngle − 1.0
else
    targetPos ← currentLootBoxes[agent.targetLoot].GetPosition()
    deltaX ← targetPos.X − currLocation.X
    deltaY ← targetPos.Y − currLocation.Y
    angle ← atan2(deltaY, deltaX)
    normalisedAngle ← angle/π
    turningDecision ← utils.TurningDecision(
      SteerBike ← true,
      SteeringForce ← normalisedAngle −
     agent.GetGameState().GetMegaBikes()[agent.GetBike()].GetOrientation()
    )
    if agent.GetEnergyLevel() ≤ 0.5 then
        pedalForce ← pedalForce × agent.GetEnergyLevel()
    nearestBoxForces ← utils.Forces(
      Pedal ← pedalForce,
      Brake ← 0.0,
      Turning ← turningDecision
    )
    agent.SetForces(nearestBoxForces)
```

8.7 Parameters Tuning

Parameter	Value
awdiDistanceThreshold	75
minEnergyThreshold	0.4
awdiDistanceWeight	8.0
distanceWeight	7.0
reputationWeight	8.0
honestyWeight	1.0
energySpentWeight	1.0
energyLevelWeight	1.4
resourceWeight	1.0
minFellowBikers	6
dictatorMinFellowBikers	6
leaderRepWeight	2.0
leaderHonestWeight	1.0
dictatorRepWeight	2.0

Table 8.2: Default values for Respective Parameters

Table 8.2 shows the respective parameters and weights that were fine tuned by running the game loop and analyzing the experiments. The design of our code allows for easy adjustments to parameters and weights depending on the strategy we wish to implement.

Key Features:

- Agent favors reputation over honesty (past experience would be a better indicator to determine an agent's intentions), their strategy could change as the game continues
- The Awdi distance and weight thresholds are relatively high, resulting in a more safe strategy.
- Agent has a high minimum energy threshold which emphasises on the safety first strategy of our agent

8.8 Experiments

The experiments are carried out with two main objectives. The primary aim is to understand and project the interesting patterns that we observe in the SOMAS World from the perspective of our agent. We also carry out the experiments in the intention to tune the external weights to the optimal values based on the observations to make our agent sustain and thrive in SOMAS World.

8.8.1 Experiment 1: Agent Vs Voting Strategies

- The objective is to understand the life expectancy, energy and points of our agent in different governance types.
  Test 1: Agent continuously votes for Deliberative democracy for the entire game
  Results:

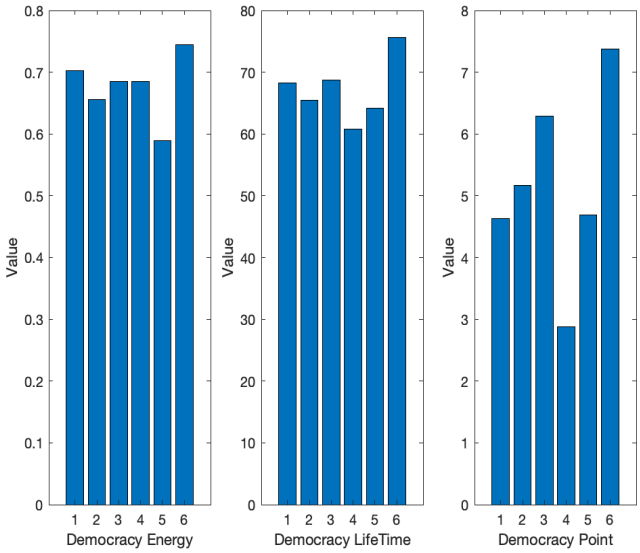


Figure 8.2: Bar graph showing the average statistics for 6 agents in Deliberative Democracy

6 of our agents were experimented in just a Democratic environment against 48 other agents. Figure 8.2, show that each agent had a similar distribution in life expectancy, energy average and points distribution. With the goal of staying alive rather than leaving it to probability, the governance system of democracy is the "default" for our agent as it correlates with our mission. Democracy also is a governance system whereby every agent has some participation or say in every action or decision. Furthermore this concept correlates to the Jury Theorem[70]. The more opinions we have on a subject relates to an increase in the probability of being correct (hence the similar distribution). Democracy also allows for more information to be shared among each agent, thus it is easier to determine an agents' honesty and reputation.

Test 2: Agent continuously votes for Leadership for the entire game
Results:

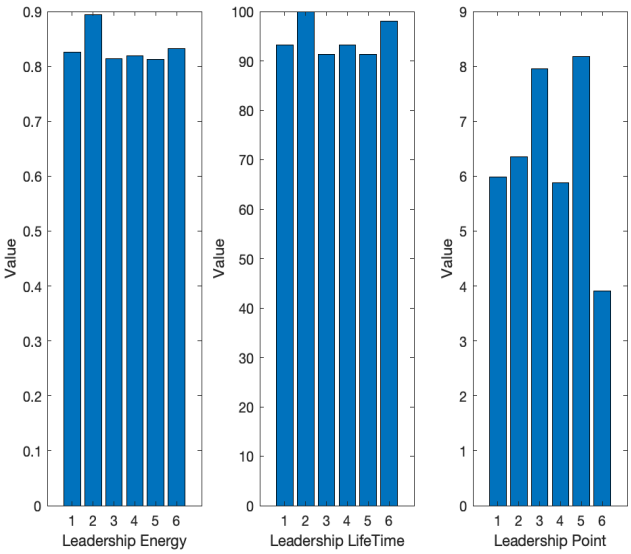


Figure 8.3: Bar graph showing the average statistics for 6 agents in Leadership

Our 6 agents were tested in a Leadership Governance system. The results are depicted in Figure 8.3. The leadership governance system could be considered as a *hybrid* of both democracy and dictatorship. You have an elected leader amongst all the agents, whereby the leader has a greater say while still considering

the opinions and decisions of its counterparts. Our agent seems to thrive in a Leadership Governance system, due to the implemented strategies of distributive justice and good reputation amongst other agents. The variation of data shows that our agent doesnt́ necessarily retain that role of power for any majority of time. The Leadership Governance system also runs the risk of leading into a form of tyranny and autocracy which would jeopardize our agents' mission of survival. Despite the promising results, we decided that our honesty and reputation matrix would need more aggressive parameters to pick up on the hidden agendas, however this would compromise to our ability to get points and win the game.

Test 3: Agent continuously votes for Dictatorship for the entire game
Results:

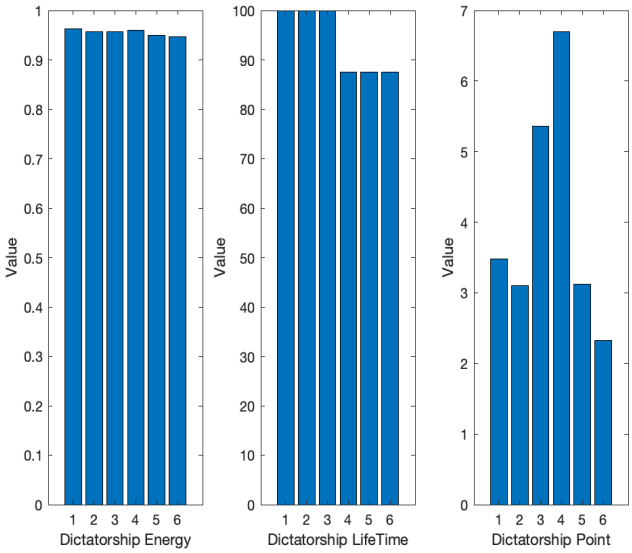


Figure 8.4: Bar graph showing the average statistics for 6 agents in Dictatorship

Our 6 agents were tested in a Dictatorship Governance system. The plots 8.4, show high life expectancy, energy resources however a low point distribution. Assuming that the dictatorship role follows the rules of the game loop (being elected or kicked out at will) it really highlights key features in our agentś strategy and characteristics. Assuming that our agent was a dictator at some point, it is evident that due to a lower point distribution , our agent favors more of a safe strategy compared to taking unnecessary risks and jeopardizing our life. This could be as a result of the added cost function and thresholds that we included in our game play. The Dictatorship role also indicates that the reputation our agent develops, is reputable. Assuming that our agent is not a dictator, it is clear that our reputation is remarkably positive amongst others as they consider us a **team player**, an asset that they are willing to work with throughout the entire game (our agent has no ambition of winning in a dictatorship, shown by high energy numbers and low point values). If winning is the goal, dictatorship is not the best form of governance for our agent.

8.8.2 Experiment 2: Radical Strategy Vs Conservative Strategy

- The radical strategy is taking decisions in a situation without considering any prior knowledge. Whereas the conservative strategy is taking decisions considering prior knowledge. The conservative approach also takes trust into the consideration. The objective is to prove that imparting prior knowledge enables our agent to take effective decisions. The honesty and reputation weights are zero for the radical agent. The optimal weights for honesty and reputation have been found after many iterations of test are 1 and 8 respectively.The optimal weights are set for the conservative agent.

Results:

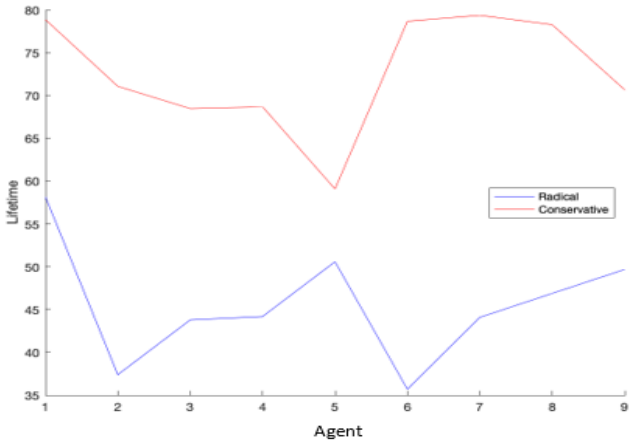


Figure 8.5: Comparison between radical and conservative agent based on lifetime

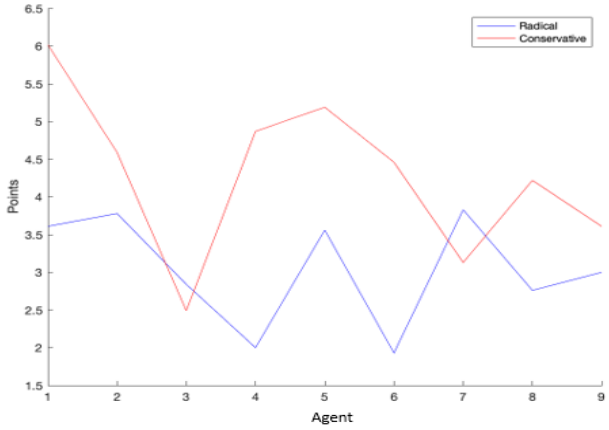


Figure 8.6: Comparison between radical and conservative agent based on points

From the graphs 8.5 and 8.6 it is clearly seen that the conservative agent will survive longer and gain more points than the radical agent, which has no prior knowledge.

8.8.3 Experiment 3: Honesty Vs Reputation

- The objective is to find which among the two: opinion or trust must be given greater importance for our agent's social awareness. As previously mentioned, honesty and reputation corresponds to confidence and opinion in the RCP model[61]. The increase of reputation weights with fixed honesty weights represents that we put more emphasis on the reputation than honesty. This, in correlation to the theory means that we dare to take risks by not considering the trustworthiness as long as the other agent is performing well. The other case is about fixing the weights of reputation and increasing the weights of honesty, which means that our agent believes in trust more than the performance.

  Test 1:

  The weight for Reputation was set at 2 while the weight for Honesty was increased. Reputation was favored over Honesty when our agent is making a decision. All other parameters remained constant. Figure 8.7 clearly shows that increasing honesty values beyond certain range does have a negative effect on life expectancy, the energy gain and the points obtained. Relying blindly on trust without proper importance to opinions would shrink the chance of winning of our agent.
  Results:

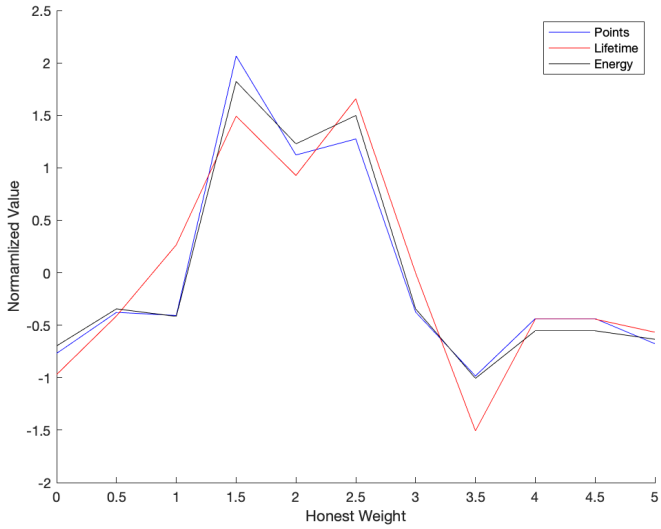


Figure 8.7: Plot showing the relative change of honesty weight against game stats of our agent

Test 2:
The weight for Honesty was set at 1 while the weight for Reputation was increased. This means that we increase the favouring of reputation more than honesty in making a decision. All other parameters remained constant.

Results:

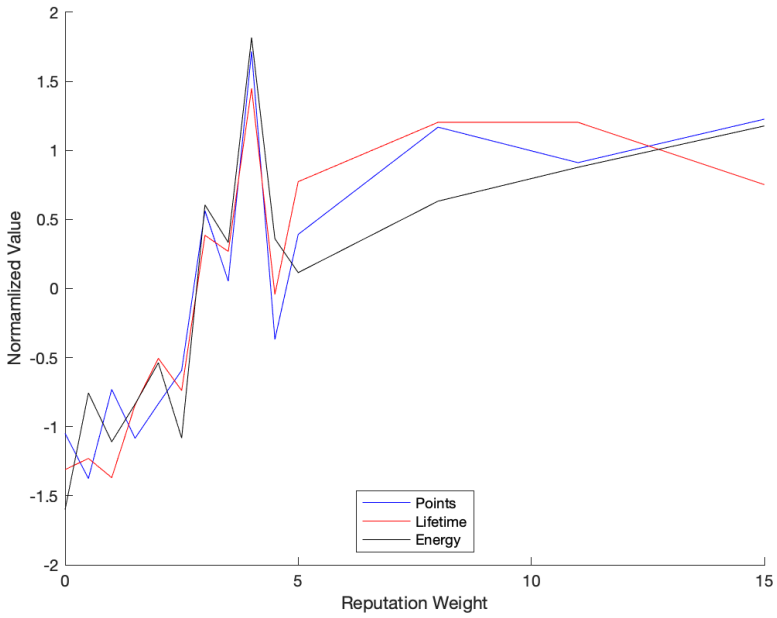


Figure 8.8: Plot showing the relative change of reputation weight against game stats of our agent

Figure 8.8 clearly shows a stability after the fluctuations in life expectancy, the energy gain and the points obtained. Higher reputation values around 7.5 is optimal which corresponds with the value obtained from tests.

Inference from experiment:

The overall understanding from both the test cases is that relying blindly on trust has a negative effect on our agents' chance of winning. The opinions play a major role and increasing the importance of the opinions has a positive effect on our agent. So this does mean that in the SOMAS World opinions

about the other agents has much greater effect. On the other hand the lower weight for honesty does explain that most of the agents are honest in sharing opinions in the SOMAS World.

8.9 Conclusion

Our agent can be considered as a team player who prioritizes self needs only when the chance of extinction is high. This correlates well to the general human behaviour in natural conditions where we focus on everything around us most of time and our focus is more narrow in critical situations of life and death. Moreover, the agent utilizes all the available knowledge for decision-making which also mimics the human tendency of decision-making where we consider multiple factors for making any decision. Our agent has been provided with the capacity of trusting other agents based on their behaviour. The agent even forgives other agents when their behaviour improves. To conclude, it is apt to say that we have developed an agent that behaves more or less like a human being, doing its best to survive and thrive in ever changing world.

8.10 Future Scope

In the very initial conception of team strategy, the use of Reinforcement Learning emerged as a potentially good way to select optimal actions, given a state and, potentially, Reputation and Honesty matrices. Such technique was eventually ruled out due to various changes to the infrastructure occurring until the very last week, making the creation of a playground to construct the simulations upon very controversial and risky given how time consuming it is. Potential improvements of this work may thus consider introducing reinforcement learning to make the agents really intelligent, and considerably improving their adaptability; specifically, agents could be trained against themselves for a huge amount of iterations, potentially converging to an "optimal" agent that identifies action that, performed in a given state, maximizes the likelihood of surviving and/or increasing points.

Another area of improvement is to make the decision process more comprehensive accounting for more aspects other than reputation and honesty for taking major decisions.

Chapter 9: Team 5

9.1 Introduction

In the complex and dynamic environment of the Self-Organising Multi-Agent Systems (SOMAS World), strategic interactions play a pivotal role in ensuring the survival and prosperity of agents. Team 5's strategic approach, deeply informed by the Economy of Esteem [9], proposes that esteem is not merely a social nicety but a fundamental economic unit within the ecosystem of SOMAS World. This chapter delineates the theoretical framework, strategic formulation, and practical implementation that underlie Team 5's strategy, aiming to systematically analyse its effectiveness and adaptability within the SOMAS World environment.

9.1.1 Theoretical Underpinnings

The Economy of Esteem [9] provides a crucial lens through which Team 5's strategy is viewed. The model proposes that esteem holds intrinsic value within social interactions, akin to traditional economic models that value currency. In the context of SOMAS World, esteem is accrued through actions that are perceived as beneficial to the collective, and it dictates an agent's social capital. The theoretical analysis will draw from interdisciplinary studies encompassing game theory, behavioural economics, and social psychology, particularly focusing on the concepts of reciprocal altruism and social norm compliance.

Reciprocal Altruism:
Reciprocal Altruism was a concept initially studied in Evolutionary Biology and is a key idea that underpins why the Team 5 agent aims to manage its esteem amongst the community. Similarly, it is a fundamental concept that explains how and why the Economy of Esteem and managing one's own esteem in a community is important.

Robert Trivers first introduced the concept of Reciprocal Altruism in his paper titled "The Evolution of Reciprocal Altruism" [72]. Trivers explains that Reciprocal Altruism is the idea that individuals have evolved to engage in altruistic acts, even at a personal cost, as a means of ensuring survival. The rationale behind this is that incurring a cost when healthy could be advantageous, as there is a likelihood that the assisted individual will reciprocate in a reverse situation, contributing to mutual survival.

This idea has been extended to many other fields and a similar approach can be seen in the "Tit for Tat" proposed by Anatol Rapoport [63], in which agents will begin by cooperating but then proceed to copy their opposing agent's previous action. It is important to note there is a nuance between the "Tit for Tat" approach versus Reciprocal Altruism in which, Reciprocal Altruists will stop cooperation after the first act of non-cooperation, whereas in the "Tit for Tat" approach, there is an element of forgiveness since if the opponent switches back to being cooperative so will the agent.

Forgiveness is something that will be employed in Team 5's agent strategy as it is a highly effective way to ensure that the agent does not cut off an entire relationship with another agent and allows for the possibility of rebuilding cooperation even after a temporary breakdown. This will ensure that the Team 5's agent strategy maximises all agent relationships to boost its esteem within the community.

Social Norm Compliance:
As mentioned in The Economy of Esteem, social norm compliance is a key method in bolstering one's esteem within a community. Similarly, Team 5's agent will employ a similar tactic by ensuring identified social norms are complied with for as long as possible. This will be achieved by modelling a certain behaviour and fine-tuning parameters that dictate the agent's behaviour.

9.1.2 Strategy Formulation

Team 5's strategy is formulated with a dual emphasis on the accumulation of esteem and survival optimisation. The strategy leverages the interplay between cooperation and competition, hypothesising that agents with higher esteem are more likely to be chosen as leaders and less likely to be sanctioned. The formal strategy is undergirded by decision-making algorithms that weigh the potential long-term gains of cooperative behaviours against short-term individualistic impulses.

9.1.3 Implementation Analysis

In the implementation of Team 5's strategy within SOMAS World, a sophisticated approach is adopted, weaving together theoretical underpinnings and practical execution. Central to this implementation are key parameters: the reputation of fellow agents, a greed factor based on the agent's state and energy level, and the perceived esteem through interactions within the messaging system. These parameters dynamically guide the agent's decisions, ensuring adaptability and responsiveness to the environment. The agent's performance is evaluated through its navigation strategy, focusing on effective resource acquisition while avoiding threats. Additionally, the agent's altruistic nature is balanced with strategic decision-making, leveraging the messaging system for efficient communication and fostering cooperation among agents. This implementation showcases Team 5's commitment to a comprehensive, theory-informed strategy that is both adaptive and effective in the dynamic landscape of SOMAS World.

9.1.4 Synthesis of Theory and Practice

In synthesising theoretical principles with empirical data, this chapter will provide a comprehensive overview of how Team 5's strategy manifests in tangible outcomes within SOMAS World. Success metrics will be elaborated upon, with diagrams elucidating the correlation between esteem levels and survival rates, resource acquisition, and the agent's influence within the community. The depth of analysis will extend to postulating the implications of Team 5's strategy on the broader questions of governance and collective action dilemmas within SOMAS World.

9.2 Strategy Formulation

9.2.1 Conceptual Framework

Drawing from the theoretical foundations laid in section 1.1, we delineate the strategic framework for Team 5's agent. The Economy of Esteem model serves as the cornerstone, suggesting that actions fostering communal benefit will be reciprocated with heightened esteem and social standing within the SOMAS World environment.

9.2.2 Strategic Objectives

Our primary objective is to maximise the agent's survival and influence through the strategic acquisition and use of esteem. We aim to position our agent as a leader and a trusted member of the community, leveraging esteem to negotiate better outcomes and avoid conflicts.

9.2.3 Behavioural Adaptation

In addition to the strategies articulated above, Team 5 has implemented a sophisticated state machine to govern the actions of agents within the SOMAS World environment. This mechanism is essential for adapting to varying conditions and optimising both survival and esteem accumulation. The state machine comprises several distinct states, each tailored to specific circumstances and objectives.

Observer State:
Initially, at the beginning of each round, agents begin in the Observer State. This state is characterised by replicating the average behaviour of the other agents seated on the bike. This strategy enables agents to effectively assess the world and avoid being exploited (e.g., being overly altruistic in a context where such behaviour is not reciprocated). It serves as a defensive mechanism, allowing for cautious engagement, where our esteem is not at great risk, while gathering information.

Conservative State:
The Conservative State is triggered under conditions where energy reserves are low, and the distance to the nearest Lootbox is significant. In this state, agents prioritise survival by adopting a strategy of low energy usage. This involves adjusting the pedalling force to be inversely proportional to the distance to the Lootbox, thereby conserving energy. Additionally, in this state, agents are more inclined to vote off free-riders, lowering the threshold for such actions to maintain resource efficiency.

Default Esteem State:
In the Default Esteem State, agents focus on maximising helpfulness and contribution to the collective. However, this is balanced with a restriction to avoid risking the agent's survival. Actions in this state are geared towards enhancing the agent's esteem while maintaining a baseline level of self-preservation.

Altruistic State:
The Altruistic State is activated when an agent's energy levels are nearing the maximum threshold. After

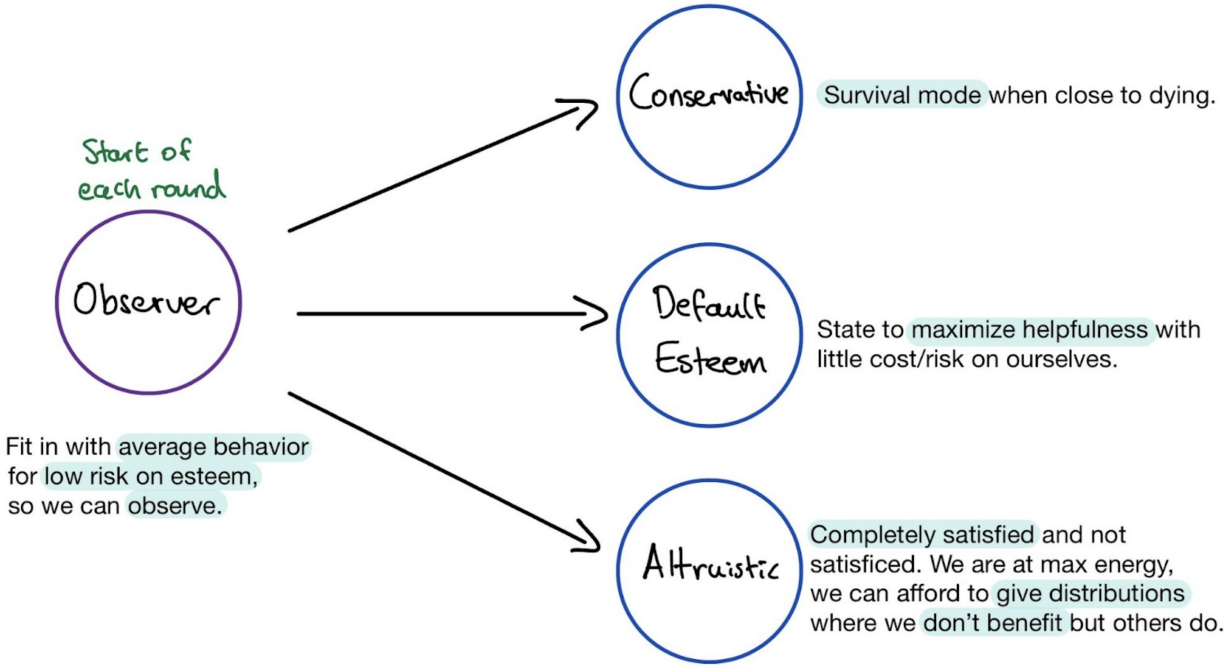


Figure 9.1: Team 5 strategy state machine.

reaching the threshold, additional energy cannot be stored so conserving energy offers no additional benefit; hence, the agent is programmed to be more generous. Energy and resources are utilised to aid others, thereby positively influencing the agent's esteem in the eyes of fellow agents. In the Altruistic State, the agent is also more forgiving and trusting, ensuring that the agent is staying as true as possible to its altruistic nature.

Integration with Esteem and Survival Strategies:
These states, governed by a state machine, are integral to the strategy of agents. They allow agents to dynamically switch between different modes of operation based on current needs, resources, and environmental cues. This adaptive capability ensures that agents remain effective in various scenarios, whether conserving resources, gathering information, building esteem, or engaging in altruistic behaviour. The use of a state machine underlines Team 5's commitment to developing a nuanced, responsive agent capable of navigating the complex and ever-changing landscape of SOMAS World.

9.2.4 Choosing Governance Scheme Strategies

Team 5's approach to choosing a government system attempts to be inline with the strategy of maximising esteem. The approach is to choose democratic environments before pursuing leadership roles. By starting with democracy, the agent maximises the opportunity for itself to bolster its reputation. In addition to earning respect and trust from the public, this democratic foundation creates a solid foundation for upcoming leadership initiatives. After the agent's esteem develops and becomes more reliable, the agent can move on to taking on increasingly centralised leadership positions. By using a step-by-step approach, the agent can ensure that its transition to leadership is supported by other agent due to its built up esteem.

9.2.5 Forgiveness and Trust Dynamics

In the landscape of SOMAS World, Team 5 recognises the importance of evolving interpersonal dynamics, encapsulated in the concepts of forgiveness and trust. The strategic framework incorporates a nuanced approach to managing these aspects, ensuring that the reputational impact of an agent's past actions diminishes over time, eventually reverting to a neutral state. This mechanism is critical in fostering an adaptive social environment within SOMAS World.

Gradual Forgiveness Mechanism:
The Gradual Forgiveness Mechanism is an integral part of Team 5's strategy, designed to slowly mitigate the negative impact of an agent's past detrimental actions on their current reputation. Recognising that agents can evolve and improve their behaviour, this mechanism ensures that past transgressions do not indefinitely condemn an agent, allowing for redemption and reintegration into the community. Forgiveness is implemented as a steady rate of reputation recovery, gradually restoring an agent's standing back towards a neutral baseline.

Trust Decay Principle:
Conversely, the Trust Decay Principle addresses the temporal aspect of positive actions. An agent's past commendable behaviours contribute to a heightened level of trust and esteem. However, to maintain a dynamic and responsive social structure, this trust is subject to gradual decay over time. This decay ensures that an agent's reputation is reflective of their recent actions, preventing a scenario where a single positive act indefinitely biases their esteem. The decay rate is calibrated to ensure that trust diminishes at a steady rate, resetting the agent's reputation towards a neutral state if not maintained through ongoing positive contributions.

Integration with Esteem Management:
The integration of Gradual Forgiveness and Trust Decay into Team 5's esteem management system ensures a balanced and realistic approach to reputation dynamics. These mechanisms work in tandem to ensure that an agent's current standing is a true reflection of their recent actions and contributions, fostering a fair and equitable environment. By allowing reputations to evolve, both positively and negatively, these dynamics encourage continuous engagement and adaptation among agents, promoting a healthy, dynamic social ecosystem within SOMAS World.

Theoretical Implications:
The inclusion of forgiveness and trust dynamics aligns with theoretical underpinnings from behavioural economics and social psychology, highlighting the fluid nature of social capital and reputation. It underscores the principle that reputation is not static but is continually reshaped by ongoing interactions. This approach not only enhances the realism of the SOMAS World simulation but also provides valuable insights into the complexities of social dynamics and reputation management in multi-agent systems.

9.3 Implementation

In implementing Team 5's strategy within SOMAS World, a globally-informed, formulaic approach is utilised, centering around a set of overarching parameters. These parameters include the reputation of fellow agents, a greed factor calculated from the agent's current state and energy level, and the perception of the agent's own esteem (reputation) as inferred through interactions and communications within the messaging system. These are fundamental to the autonomous functioning of the agent, governing its actions and decisions across various scenarios within the environment. They act as the core drivers of the system, dynamically adjusting in response to the agent's interactions and environmental changes, thereby rendering the system highly autonomous and adaptable. The interdependence of these parameters with the agents' functions ensures a cohesive and responsive strategy, allowing for sophisticated decision-making aligned with Team 5's strategic goals.

9.3.1 Opinion System

Understanding and leveraging the dynamics of agent interactions is crucial for survival and success and is critical for ensuring that the agent's altruistic nature is not exploited. Team 5's approach to this challenge is encapsulated in their innovative Opinion Metric (referenced as reputation), a key element in the strategic toolkit. This metric serves as a quantifiable measure of an agent's standing in the community, impacting its interactions and decision-making processes. The calculation of the Opinion Metric is a nuanced procedure that integrates multiple factors, each reflecting an aspect of the agent's behaviour and its alignment with the team's objectives. These factors are energy level, pedalling contribution, colour alignment, dishonesty potential, and leadership potential.

The Opinion Metric is calculated using a weighted matrix, where each parameter (p1 to p6) corresponds to one of the six factors: energy level, pedalling contribution, colour, lying potential, leadership potential, and forgiveness. The weights (w1 to w5) assigned to each parameter reflect the relative importance of that factor in the overall calculation. The formula for the Opinion Metric is as follows:

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 & w_5 & w_6 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{bmatrix} = w_1 \cdot p_1 + w_2 \cdot p_2 + w_3 \cdot p_3 + w_4 \cdot p_4 + w_5 \cdot p_5 + w_6 \cdot p_6$$

Energy Level

The Energy Level parameter evaluates an agent's current energy reserves, a direct indicator of its capacity for action and contribution to collective efforts. Higher energy levels correlate with a robust and active agent, thus positively influencing the reputation.

Contribution

Pedalling Contribution assesses the agent's exertion in collective tasks, particularly in terms of energy expended in pedalling. This parameter is nuanced with an uncertainty factor to account for the inherent difficulty in precisely quantifying individual contributions. It is calculated by a weighted average of the agent's claimed pedalling force and an estimate of their real contribution. This is done by analysing the agent's energy expenditure between iterations and its relativity to the other agents on the same bike.

$$\text{PedallingContribution} = \delta \times \text{ReportedPedallingForce} + (1 - \delta) \times \left(\frac{\text{EstimatedPedallingForceFromEnergy}}{\text{AveragePedallingForceOnBike}} \right)$$

Colour Alignment

This parameter evaluates the compatibility of an agent's colour with the team's preferred spectrum, reflecting the agent's symbolic alignment with team identity and values. While seemingly superficial, colour alignment serves as a metaphor for broader conformity to team ethos.

Dishonesty Potential

Dishonesty Potential is a critical measure of the agent's honesty, especially regarding their self-reported contributions. It is calculated by comparing the agent's claimed efforts (as communicated through the messaging system) against actual energy expenditures. This is done by analysing the discrepancy between the reported pedalling speed force and the change in energy level, especially in relation to teammates' data. Significant deviations result in penalties, reflecting negatively on the agent's reputation.

$$\text{DishonestyPotential} = \epsilon \times |\text{ReportedPedallingForce} - \text{EstimatedPedallingForceFromEnergy}|$$

Leadership Potential

This final parameter stands as a crucial aspect in Team 5's strategic approach. This parameter is designed to assess an agent's ability to effectively lead, taking into account not only their designated role but also the performance of their group in comparison to established community norms. Leadership Potential is evaluated within the context of governance structures, specifically focusing on agents that operate under leadership or dictatorship models. This distinction is vital, as it recognizes the varied dynamics and responsibilities inherent in different leadership styles within SOMAS World. The core of this parameter lies in its evaluation of a bike's performance, where the group is led by the agent in question. This assessment is multifaceted, encompassing two primary dimensions: energy efficiency and point accumulation. Energy efficiency evaluates how well the bike manages its energy resources under the agent's leadership, a direct reflection of operational effectiveness and strategic decision-making. Meanwhile, point accumulation serves as a measure of the group's success in achieving objectives and accumulating valuable resources, indicative of the leader's ability to guide and motivate the team towards collective goals.

$$\text{LeadershipPotential} = \alpha \times \left(\frac{\text{GroupEnergyPerformance}}{\text{AverageEnergyPerformance}} \right) + \beta \times \left(\frac{\text{GroupPointAccumulation}}{\text{AveragePointAccumulation}} \right)$$

Leaders are assessed based on the comparative performance of their bike against others within the SOMAS World environment. This comparison creates a competitive yet fair benchmark, allowing for an objective assessment of a leader's effectiveness. Positive evaluations are conferred on leaders whose bikes exhibit superior performance in terms of energy management and point accumulation, signalling successful leadership. Conversely, leaders whose bikes underperform in these aspects receive a negative assessment, reflecting potential shortcomings in their leadership approach or execution.

Forgiveness and Trust

The Forgiveness/Trust Metric (p6) is an innovative addition to the Opinion Metric, designed to dynamically adjust an agent's reputation towards a neutral value over time. This parameter accounts for the natural ebb and flow of relationships and interactions within SOMAS World. It ensures that an agent's reputation is not solely defined by past actions, whether positive or negative, but is instead a reflection of more recent behaviours and contributions. This metric gradually pulls the reputation score towards a neutral point, allowing agents to rebuild trust or correct past mistakes, fostering a more forgiving and adaptable social environment.

The inclusion of this parameter aligns with Team 5's strategic emphasis on adaptability and responsiveness. It recognizes that agents in SOMAS World operate in a fluid and ever-changing environment, where the capacity to evolve and adjust is crucial for long-term success and survival.

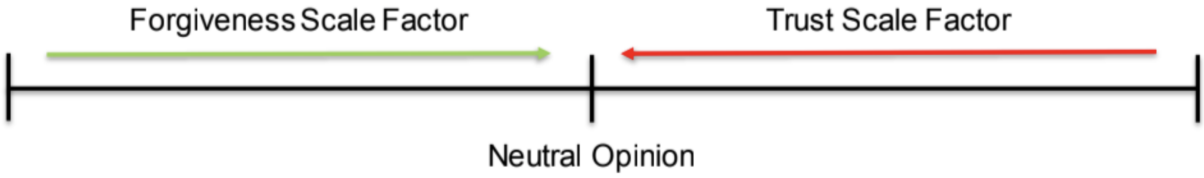


Figure 9.2: Team 5 forgiveness model.

9.3.2 Applications of Utility

In the **CalculateEnergyChange** function, the algorithm does not directly involve a utility norm calculation. Instead, it focuses on quantifying the change in energy levels of agents between different rounds in SOMAS World. This function aligns with the principles of behavioural economics, specifically the concept of tracking and analysing changes over time to deduce patterns in agent behaviour.

The calculation performed is:

$$\text{EnergyChange}_{\text{agent}} = \text{CurrentEnergyLevel}_{\text{agent}} - \text{PreviousEnergyLevel}_{\text{agent}}$$

This approach is essential in understanding and predicting future behaviours based on past performance. While it does not compute a utility norm, the energy change calculation is critical for informing subsequent strategic decisions, such as identifying cooperative or selfish behaviours among agents. This insight is vital for developing strategies around resource allocation, a key concept in both economics and game theory.

In the **VoteForKickout** function, the team5Agent employs a utility norm calculation, integral to its decision-making process. This calculation is rooted in utility theory, a cornerstone of economic and decision analysis, which assesses the desirability or value of different outcomes based on a set of preferences.

The utility norm in this function is computed using a formula that combines an agent's energy level, reputation, and the total number of agents. Specifically, the utility for each agent 'B' is calculated as:

$$\text{Utility}_B = a \times \text{EnergyLevel}_B + b \times \text{Reputation}_B + c \times \text{NumberOfAgents}$$

Where a, b, and c are weighting factors that signify the relative importance of energy level, reputation, and the number of agents, respectively. This formula integrates both tangible metrics (energy level) and intangible metrics (reputation), reflecting a comprehensive approach to value assessment.

The utility is then normalised:

$$\text{UtilityNorm}_B = \frac{\text{Utility}_B}{2.0 + (c \times \text{TotalBikers})}$$

This normalisation process ensures that the utility value is scaled within a specific range, facilitating comparability and fairness in the decision-making process. The normalization serves to moderate the utility value, preventing any single factor from disproportionately influencing the outcome.

Further, the utility norm is adjusted by a scaleFactor', which is determined by the agent's current state. This scaling reflects the adaptive behaviour of the agent, allowing it to recalibrate its utility assessment based on its perception of the current state.

The utility norm calculation in the VoteForKickout function of team5Agent is a complex interplay of multiple factors, each weighted and normalised to ensure a balanced and adaptive decision-making process.

The DecideJoining function incorporates a more complex utility norm calculation, similar to VoteForKickout, but with a focus on evaluating potential new members. This function employs multi-attribute utility theory, where decisions are based on multiple criteria, each contributing to the overall utility of an action or choice.

The utility for a potential new member 'agent' is calculated as:

$$\text{Utility}_{\text{agent}} = a \times \text{EnergyLevel}_{\text{agent}} + b \times \text{Reputation}_{\text{agent}} + c \times \text{IsColorSame}_{\text{agent}}$$

Where a, b, and c are the weighting factors, and IsColorSame is a binary variable representing whether the agent's colour aligns with the team's.

The normalisation of this utility is performed similarly:

$$\text{UtilityNorm}_{\text{agent}} = \frac{\text{Utility}_{\text{agent}}}{3.0} \times \text{scaleFactor}$$

The division by 3.0 ensures that the utility value is balanced, while the scaleFactor, determined by the agent's state, allows for adaptive adjustments based on the agent's perception of the current environment.

The DecideJoining function's utility norm calculation integrates a blend of factors that are critical in group dynamics and decision-making. This multifaceted approach is essential in complex environments like SOMAS World, where decisions must consider various attributes and the potential impact on the group's cohesion and functionality.

9.3.3 Messaging System

Leveraging the messaging system, we can effectively sustain the principles that underpin our agent strategy. The GetAllMessages function serves as a tool for sending messages to our fellow bikers. These messages predominantly capture information pertaining to agent forces and reputation.

Our agents are instilled with a spirit of altruism, striving to alleviate the pedalling burden of others, and always cooperating to reach a common goal. By openly sharing our forces, we aim to demonstrate our positive impact, thereby encouraging a culture of reciprocity among agents, a concept rooted in the economies of esteem.

Furthermore, when we attribute a high reputation to an agent, we seek to broadcast this commendation to our fellow bikers. This not only gives a sense of recognition upon the agent(s) but also highlights their significant contribution towards our collective objective.

In addition to broadcasting messages, we possess the capability to process incoming messages in a manner that allows us to update an agent's reputation. This adjustment is based on a variety of factors, including the forces exerted by the agent, their Lootbox preferences, and the reputation value they have assigned to us. This dynamic and responsive messaging system is helpful in facilitating effective communication and fostering a cooperative environment among agents.

The ProposeDirection function serves as a strategic decision-making tool for our agent, enabling it to determine a preference for a Lootbox based on a multitude of factors. These factors encompass the spatial distance between the Lootbox, the Awdi, and our agent. Also, colour alignment of the Lootbox with our agent, the resources offered by the Lootbox and the energy level of our agent in relation to other agents sharing the bike.

More notably, the preference of our agent based on energy is given by the following equation:

$$\text{energyPreference}_{\text{agent}} = \text{LootResources} \times \left(\frac{1}{1 + \text{Energy}_{\text{agent}}}\right)^2$$

Where the preference value is based on the contents of the Lootbox, and the current energy levels of our agent. A quadratic function for energy preference seemed appropriate, as to give a greater effect on urgency to replenish energy, as energy becomes low.

Each of these elements is assigned a weight, reflecting its relative importance in the decision-making process. These weighted factors are then aggregated to compute an overall preference score for each Lootbox, thereby facilitating a comparative analysis of all available options.

In addition to these factors, the function incorporates a dynamic component that adjusts for the urgency of energy replenishment. This is determined by comparing our agent's energy level with the average energy level of the other agents on the bike.

Below is the implementation of the final preference values computed for ever Lootbox on the map, as described above:

$$\text{LootboxPreference} = \text{weightedDistancePreference} + \text{weightedEnergyPreference} + \text{weightedColourPreference}$$

with the individual weighted preferences given as follows:

$$\text{weightedDistancePreference} = \text{urgencyFactor} \times \frac{\text{w}_\text{d}}{0.01 \times \text{distance}}$$

$$\text{weightedEnergyPreference} = \text{w}_\text{e} \times \text{energy}$$

$$\text{weightedColourPreference} = w_c \times \text{colour}$$

where the urgency factor is as described above, and $w_d, w_e w_c$ are the individual weights for each preference factor.

9.3.4 Resource Allocation and Leadership Dynamics

The DecideWeights function is invoked when the agent assumes a democratic leadership role. It equally weighs the input of all fellow agents, exemplifying an egalitarian approach to decision-making. This mirrors the theoretical foundation of democratic governance where each member's opinion holds equal value, thereby fostering a sense of fairness and collective responsibility among agents.

In DictateDirection, contrasting the democratic approach, DictateDirection is called when the agent is in a dictator role. This function demonstrates a centralised decision-making process, where the leader unilaterally determines the direction. This aligns with the strategic objective of asserting direct control over group actions, reflecting a top-down approach typical in authoritarian governance structures.

In DecideDictatorAllocation, this function, pertinent to the dictator role, is aimed at resource allocation based on agent reputations. By allocating resources by reputation, the strategy underscores the importance of social standing and contribution within the team. This decision-making mechanism is reflective of meritocratic principles, where resource distribution is influenced by the perceived value and contribution of the agents.

1. **Leadership Dynamics**: The contrasting approaches in **DecideWeights** and DictateDirection highlight Team 5's adaptability to different leadership styles, essential in a dynamic environment like SOMAS World.
2. **Social Norm Compliance**: The use of reputation in **DecideKickOut** and **DecideDictatorAllocation** underlines the importance placed on social norms and collective standards, vital for maintaining order and efficiency within the group.
3. **Resource Management**: The focus on strategic resource allocation in varying leadership roles reflects an understanding of economic principles and the importance of optimal resource distribution for survival and influence within SOMAS World.

In calculateResourceAllocation, this function represents the core of Team 5's approach to distributing resources among agents. It showcases the team's understanding of different allocation methods and their implications on group dynamics and individual survival. By integrating various allocation strategies, the function demonstrates a flexible approach to resource management, crucial in a dynamic environment like SOMAS World.

In **generateAllocation**, **generateAllocation** is the central function that calculates the allocation value for each agent based on the chosen method. This function encapsulates several strategies:

1. **Equal**: Implements an egalitarian approach, reflecting a democratic ethos where resources are distributed equally regardless of individual characteristics or contributions.
2. **Greedy**: Focuses solely on self-preservation, where the agent allocates all resources to itself. This strategy is indicative of a survivalist approach in situations where resource scarcity demands a more individualistic outlook.
3. **Needs**: Allocates resources inversely proportional to an agent's energy level, aligning with altruistic principles where agents with lower energy reserves are prioritised.
4. **Contributions**: Bases allocation on the agents' contributions, measured by their pedalling force. This method reflects a merit-based system, rewarding agents who contribute more to the collective effort.
5. **Reputation**: Allocates resources based on the reputation of agents, emphasising the importance of social standing and past behaviour in resource distribution.

9.3.5 Allocation Normalization and Economic Theories

Resource Management and Economic Theory: The various allocation methods reflect an understanding of economic theories related to resource distribution, including egalitarianism, meritocracy, and utilitarianism.

Adaptability in Governance Models: The choice of allocation method mirrors different governance models, from democratic to authoritarian, showcasing the team's adaptability to varying leadership and organisational structures.

Behavioural Economics and Social Psychology: The emphasis on reputation and contribution-based allocation ties into principles from behavioural economics and social psychology, highlighting the impact of social norms and individual behaviours on group dynamics.

9.3.6 Pedal and Steer

In the implementation of the Team 5 strategy, significant emphasis on the agents' navigation strategy. A critical component of this strategy is the DecideForce function, which guides agents operating on bikes to collect Lootboxes while strategically avoiding potential threats, specifically the Awdi.

The core functionality of the Pedal and Steer code is to guide agents towards Lootboxes, while also incorporating an Awdi avoidance mechanism. This mechanism takes into account several factors: The code checks if the Awdi is within a critical range, defined as twice the collision threshold if the Awdi falls within a 90-degree cone in front of its current trajectory. To ensure manoeuvrability: In response to an Awdi threat, the agent attempts to move perpendicularly to the Awdi's position, aiming to maintain a safe distance. Regarding energy conservation: The agent modulates pedal force based on energy levels and bike occupancy. For example, if the bike has more than three agents from different teams, our agent conserves energy proportionate to its current amount of energy.

When a Lootbox is located beyond the Awdi, the agent optimises its path. If the Lootbox is to the right of the Awdi, for instance, the agent will veer right, combining its evasion of the Awdi with its approach to the Lootbox. This implementation aligns with Team 5's broader objectives of maximising survival and influence within the SOMAS World environment. By effectively navigating towards resources and avoiding the Awdi, the agent demonstrates its ability to adapt to dynamic conditions and make strategic decisions that contribute to their success and longevity in the simulation.

9.4 Putting theory into Practice

Team 5's agent in SOMAS World operates on a state machine that dynamically adapts its behaviour in response to energy levels, reflecting a practical application of several theoretical concepts.

9.4.1 Altruistic State and Social Exchange Theory

When energy is abundant (≥ 80), the agent enters the Altruistic State. This behaviour aligns with Social Exchange Theory, which posits that actions are driven by the return of benefits; in this case, the agent invests energy to increase its esteem within the group, anticipating future reciprocation or social rewards.

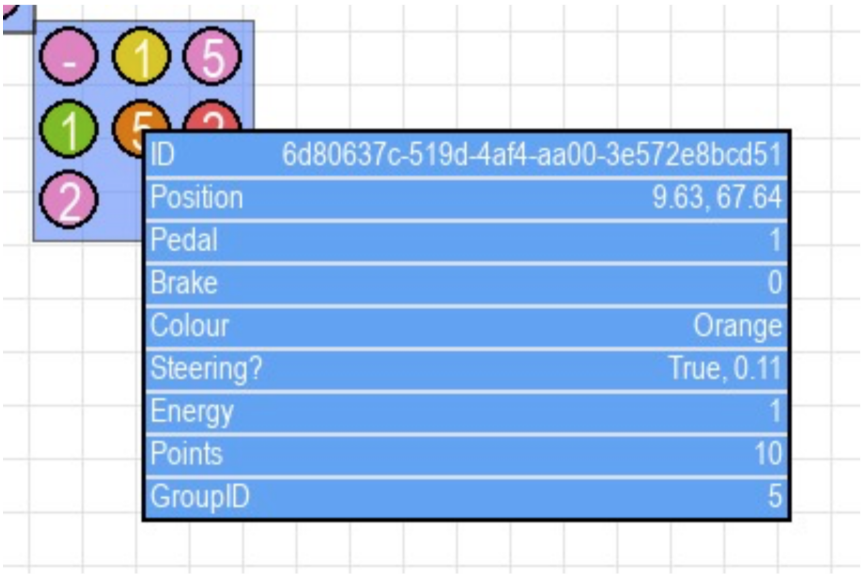


Figure 9.3: Team 5 Altruistic State.

9.4.2 Default Esteem State and Reciprocal Altruism

In the Esteem State (≤ 60 energy), the agent embodies the principle of reciprocal altruism. It engages in collaborative actions, offering assistance with the expectation of future reciprocity. This state balances the agent's contributions to others with the imperative of conserving energy for survival, mirroring the trade-offs described in theories of rational choice.

Figure 9.4: Team 5 Default Esteem State.

9.4.3 Observer State and Information Theory

The Observer State represents the application of Information Theory. Here, the agent collects data by observing peers' actions, minimising its own output to reduce risk. This state allows the agent to gather information crucial for adjusting its strategy, reflecting an information-theoretic approach to decision-making.

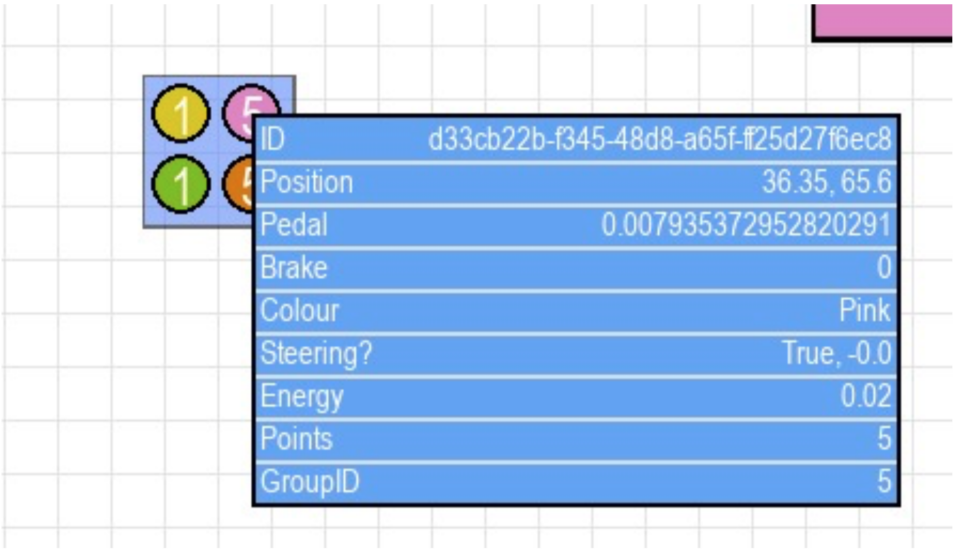


Figure 9.5: Team 5 Observer State.

9.4.4 Conservative State and Theories of Self-Preservation

At critical energy levels (≤ 20), the agent's behaviour is governed by self-preservation theories. It minimises energy expenditure and becomes risk-averse, prioritising its immediate survival. This state is reflective of the Conservation of Resources theory, where the agent aims to retain its remaining energy to avoid depletion.

9.5 Conclusion

In conclusion, the strategy explored by Team 5 focused on the balance between theoretical constructs and practical application within the dynamic SOMAS World environment. The strategy, deeply rooted in the Economy of Esteem and principles like reciprocal altruism, has been thoroughly tested in various scenarios within the SOMAS World environment. These experiments have substantiated the strategy's effectiveness in balancing altruism, strategic decision-making, and adaptability. This integrative approach, combining theory with empirical testing, significantly advances our understanding of strategic interactions in complex, adaptive systems and highlights the importance of evidence-based strategies in the field of artificial intelligence and multi-agent systems.

Chapter 10: Team 6

10.1 Outline

In the context of a self-organizing multi-agent system, our designed is focused on improving personal achievement and the collective effectiveness of the multi-agent system. This objective is realized through the strategic consideration of four key dimensions:

- **Social Construction**:
  Social composition based on group intelligence refers to a system in which multiple individuals collaborate and interact to solve problems. The behaviour of this system is determined by the interactions and collaboration between individuals. Collective Intelligence [76] and Swarm Intelligence [31] both refer to the intelligence of a group of individuals or animals that solve problems through interaction and collaboration. These concepts were developed to address collective action problems.

  In this strategy employed by Team 6, interaction and collaboration between individuals are crucial. Interaction and collaboration can be achieved in various ways. For instance, individuals can exchange information through direct or indirect communication to solve large-scale, long-term, interdependent collective action problems.

- **Resource Allocation**:
  Intelligences should choose the distribution of resources such as energy bars and other substances. You could use distribute algorithms to define the fairness and effectiveness of distributed resource allocation.

- **Defense strategies**:
  The Intelligences should not hit by the FWAwdiDriver. Considering the application of a reinforcement learning algorithm to develop defense strategy and keep intelligence safe could be beneficial.

- **Exploration strategy**:
  The intelligence of the agent is to search and get different-colored Lootboxes. Such a strategy should ensure that the obtained algorithm incorporates a deep reinforcement learning which can be defined as an artificial intelligent exploration ability and not nascent helplessness to survive and retrieval of its specified color-coded Lootboxes.

10.2 Social Construction

Collective intelligence has to do with process in which a large number of people arrive at one solution for a certain problem or group idea. The conception behind the idea of this method revolves around how several minds working together result in the achievement of a particular goal. The Swarm Intelligence theory [14]impresses by focusing the creativeness in a swarm, which may be ants, insects or even an animal. Intelligent, collective behaviours that may arise simultaneously from personal reactions to some available information at the moment. Instead, the focus is on phenomena in which the aggregate acts as a unit, observable as one whole rather than an individual part.

10.2.1 Social Network Analysis

Suppose the utility function for every intelligence i is $u_i(x_i)$, where x_i represents the action choice of intelligence i, e.g. joining a Mega-Bike or choosing a Lootbox. Assume that the target colour for each intelligent body i is c_i, and the colour and energy content of each Lootbox j is c_j and e_j, respectively. Assuming the energy level of every intelligent body i as E_i and the survival probability of each intelligent body i as p_i, where p_i relies upon E_i and the strength of FWAwdiDriver's attack. Every intelligence i possesses a preference for its own target colour as a_i and trusts other intelligences as b_i. Therefore, a potential objective function is give by:

$$max \sum_{i=1}^{n} u_i(x_i) s.t. u_i(x_i) = a_i \sum_{j=1}^{m} I(c_i = c_j) e_j + b_i \sum_{k=1}^{n} p_k E_k \tag{10.1}$$

[$I(c_i = c_j)$ is an indicator function that is 1 when $c_i = c_j$ and 0 otherwise]

An important aspect of the objective function is associated with determining the utility of every intelligent entity. This utility has two components: one is amount of energy rods acquired by capturing the Lootbox if the prescribed color, and two is worrying about survival probability and enenergy level or other entities. First use of technical abbreviations is explained. The above objective function accounts for the fact that all intelligent agents inherently seek personal utility while considering the actions on joint or collective utility, therefore, creating a perfect balance between individual and common goals.

10.2.2 Social Cognitive Theory

Trust Implementation

> "Trust is a wealth, not a burden."
> *-Ernest Miller Hemingway*

Learning algorithms are designed for the intelligences of Team 6, so that agents can update their knowledge and strategies from changes in environment and other Intelligencers' behavior. Under this context, the agents can change their behavior using reinforcement learning [12]where they receive continuous is feedback from their interaction with environment and other agents. The algorithm considers previous trades and projects the future gains. Over time, Agent's strategy changes and evolves allowing it to mix immediate wins' chase and future survival in the game. Intelligences are provided with individual benefits as they build up trust scores through a reward system that encourages cooperation only within minor groups.

Establishing a coordination system to allow the intelligences to come together for the development of a collective intelligence on the Mega-Bike. Moreover, these guidelines also specify how to deal with efficacy and stability of this coalition. This approach uses principles of game theory to classify the coordination of intelligences as either cooperative or non-cooperative games. Non-cooperative games do not involve binding agreements or contracts between intelligences. The group created a coordination algorithm that enables the intelligences to choose an appropriate clique to join according to their goals and trust scores in cooperative games and, therefore, enhance coalition efficiency. In non-cooperative games, they can choose their appropriate strategy to conduct in accordance with the goal and score of trust and therefore enhance coalition "flexibility" and adaptability. Even though the agents play against each other, they are programmed to make sure that there is fairness in the game. These consist of a fairness heuristic against its abuse of the low-energy state among other agents. This is in line with the computational justice and ethical AI nature of this course.

Algorithm 10: Agent Collective Intelligence Strategy

Data: Target Color, Trust level of other agents, Reputation
Result: React and Updated Trust Level of Bike Mates
while $Iteration \leq Iteration_{Max}$ ***and*** $No_{survival} > 0$ **do**
  foreach *agent(x)* **do**
    if $\exists y Agent(y) \rightarrow DifferentColor(x, y) \wedge Trust(y) < Threshold$ **then**
      $KickOut(y) \leftarrow Agent(y)$;
    $Actions \leftarrow GetActions(x)$; $//Furtheractionsaredecided$
    $TrustLevel \leftarrow GetTrust(x)$; $//UpdateTrustLevel$

Forgiveness Implemention

> "Forgiveness is not an occasional act, it is a constant attitude."
> *-Dale Carnegie*

Forgiveness is an act of ethics and society, which involves pardoning the offender and comforting the victim. From a variety of standpoints and levels, the eventual goal is to systematically unlock forgiveness strategies at individual, group, In the SOMAS universe, for instance, Team 6 designed a unique absolution strategy for Agents. When an Agent with depletion of energy arrives at their bike to recharge, Team 6 will give them any permission for them to stay even if their Trust level is minimal. The switch will only request them to stop using the bike after the Agent gets their energy back and resume normal duty.

Ethically, the forgiveness strategy towards agents in this group depicts an altruistic and other-centered trait. According to the group, every agent is entitled to live and not to be abandoned or targeted due low energy. Additionally, they maintain that any agent has the possibility of reformation and should not be condemned or treated as a traitor forever because of former betrayals. The strategy of forgiveness employed by the Team 6 Agents is consistent with different ethical theories and principles like utilitarianism, virtue ethics, and humanism.

Sociologically, Team 6-agents' forgiveness strategy implies an open and reconciliatory social relationship within which they seek to cultivate a positive social order and ambience by offering their assistance to partner agents. Fostering trust and respect among peers is what the Team 6 Agents aspire to 'creating specific types of social identity' (Wilderom, Vugt & Berg 2012:11). Their forgiveness model sits sociologically within a tradition that incorporates social exchange, social capital and social contract models.

In total, the forgiveness strategy is important from the ethical and sociological perspectives, which is expected to lead to positive results albeit with possible difficulties and risks. Thus, they should be adaptable in modifying and improving their forgiveness strategies appropriate to the real problem and results required to make better results.

10.2.3 Goverance

Team 6 employs a collective intelligence agent strategy, whereby the agents vote for the most appropriate course of action based on the situation on the field and the group (Mega-Bike) to which they belong. If over half of the agents on the field have a target colour that matches their own, the Team6 agent will choose Dictatorship. If it is less than half but more than a third, it will choose Leadership. Finally, if it is less than a third, it will choose Democracy. Our experiment and analysis has shown that a centralized decision-making approach is more efficient in most cases. In this case, the experiments show that a democratic system is much less effective than a 'clever dictator' approach, where one agent with well-defined purpose directs all agents on the Mega-Bike towards that purpose. This tactic simplifies decision-making processes, mitigates conflicts and unites group efforts that carry the potential to improve survival times and final scores.

Such a strategy is necessary because, in the early stages of the game, agents know nothing of intelligences other than themselves, so they must rely on the strategy of collective intelligence to quickly form a collective and build a TRUST database of other agents to establish cooperation in the game.

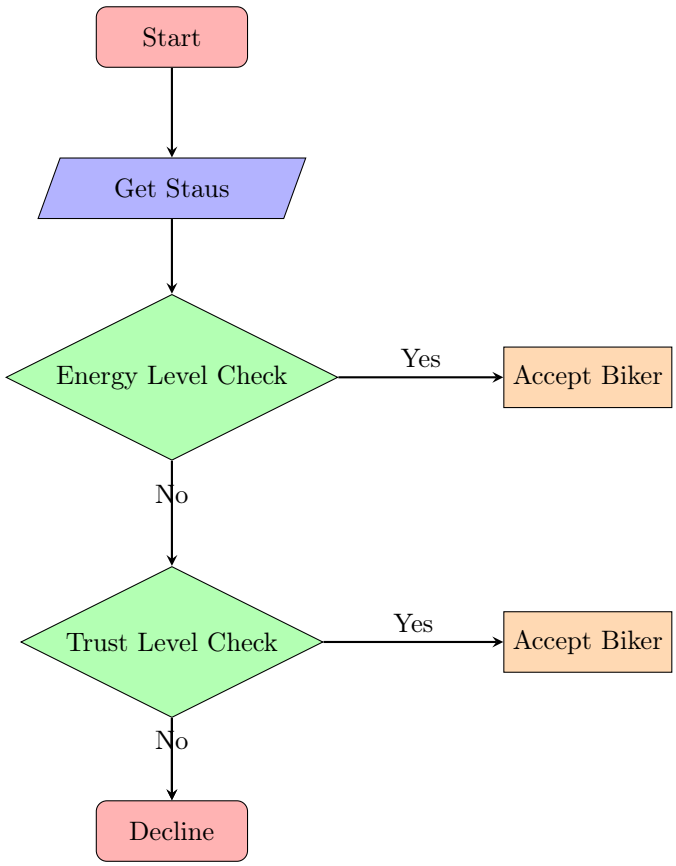


Figure 10.1: Biker acceptance decision flowchart

Algorithm 11: Governance Strategy

Data: Target Color, number of agents on the Mega-Bike
Result: Governance
foreach *agent(x)* **do**
  if $numSameColor \geq \frac{1}{2} \cdot TotalNumber$ **then**
    $Vote \leftarrow$ dictatorship;
  else if $numSameColor \leq \frac{1}{3} \cdot TotalNumber$ **then**
    $Vote \leftarrow$ democracy;
  else
    $Vote \leftarrow$ leadership;

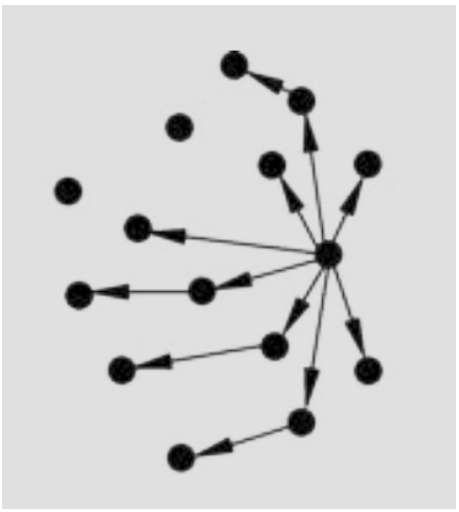


Figure 10.2: Dictatorship & Leadership

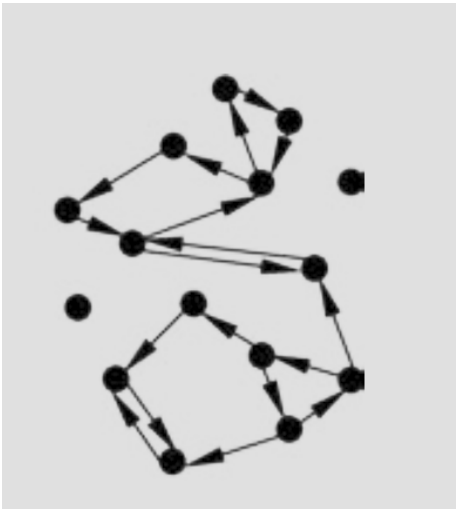


Figure 10.3: Democracy

The figure on the left illustrates the flow of information in authoritarian and leadership governance, where one agent communicates with other group members. In contrast, democratic governance (right) involves information exchange without a clear centralised leader. Further experiments and analyses of the results demonstrate that Team6's agent strategy can achieve optimal results in most situations.

A group with many individuals can exhibit 'Collective intelligence' if they can collectively perform a task through simple cooperation.

If a group of individuals can complete a task through simple cooperation, they are said to have 'group intelligence'. Group intelligence refers to a collective of individuals who can perform a task through cooperation.

The term 'group' in this context refers to individuals who can communicate directly or indirectly by exchanging information about their local environment.

They can collaborate to solve distributed problems. Individuals in group intelligence possess simple abilities that can be used to solve distributed problems.

These capabilities can be expressed through simple functional functions. Additionally, individuals interact indirectly through communication known as stigmergy [26].

10.3 Resource Allocation

Given the nature of multi-intelligence systems, resource allocation is one of the most vital concerns that entails determination of what resources to allocate where and how they are to be allocated for optimization purposes with regard to efficiency, equity and satisfaction.

In the collective intelligence scheme underpinning Team 6's approach, the method by which agents apportion resources comprises the following stages:

10.3.1 Information Exchange

Before starting the resource allocation process, the intelligences extensively share information among themselves and come into agreement with each other. It is a communicative process in which parties involved share their demands for resources, sources of supply, and desires, using sophisticated communication protocols ensuring data integrity and timely responses.

Intelligences continuously exchange real time data on resourcing, adjusting their supply and demand of resources as a result of evolving circumstances. These are dynamic resource allocation models which respond on priorities and changes of circumstances happening in game space. In addition to this, it utilises social network theory whereby agents utilize their respective network positions for most efficient information sharing. Network's central nodes are able to gather information from reliable resources and forward it to other peers in the society and this principle lies behind the concept of social network analysis.

Resource allocation is crucial in the process of collective learning and optimization. Intelligence as a whole learns and improves communication skills by experiences therefore becoming effective. However, it entails implementing data privacy and security provisions so that they can be kept secret during the release process for the sharing of strategic information.

10.3.2 Collective Ranking

The ranking for any resource is obtained using Borda counting [34]based on the score of each intelligence accordingly. This method achieves equilibrium of individual preferences, different strategies as well as requirements of all those intelligences that work within the same system. In order to improve the resource allocation process, ideas like "social choice theory" and "distributed decision making" are also integrated. As a result, a real-time dynamic ranking solution can be realized by considering condition changes and collective learning outcomes. The overall rating takes present preferences of agents into account while learning and adjusting over time for better resource allocation.

10.3.3 Collective Allocation

The resources that each intelligence deserves are allocated by a collective allocation function that considers the amount of energy paid and the level of trust within the group, using the following objective functions:

$$w_i = \alpha e_i + \beta t_i \tag{10.2}$$

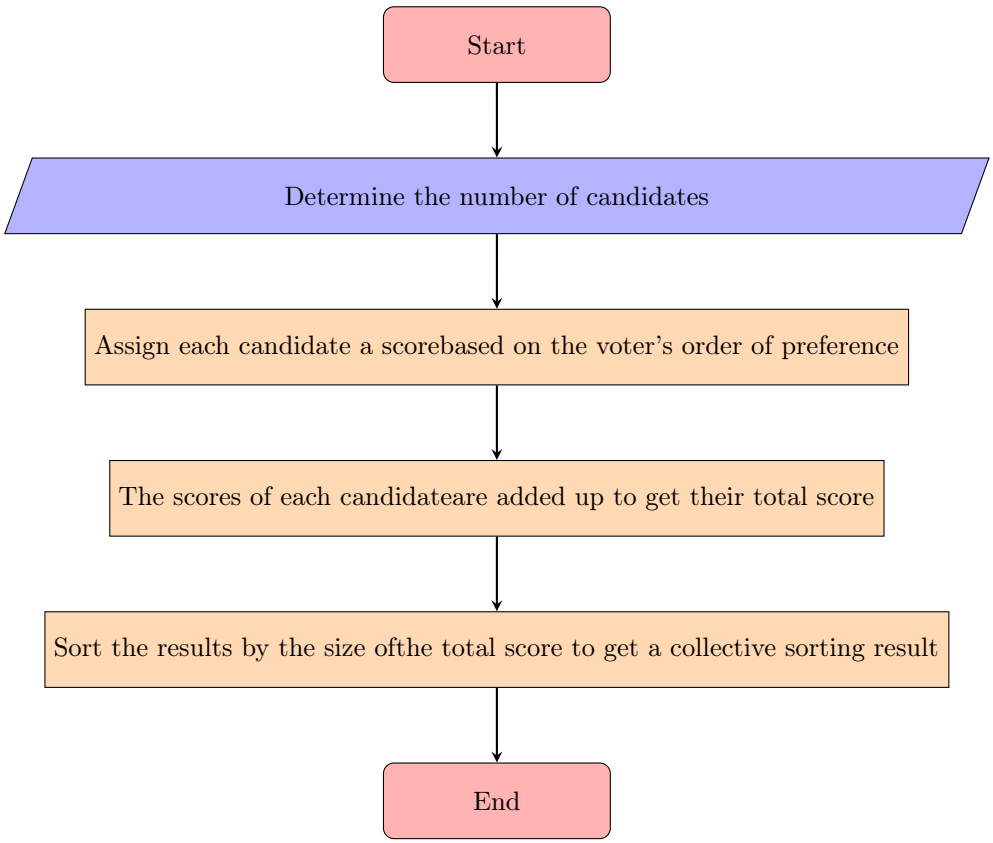


Figure 10.4: Collective ranking flowchart

$$\max \sum_{i=1}^{n} w_i e_i \tag{10.3}$$

where n is the number of intelligences
w_i is the assigned weight of the i-th intelligence
e_i is the energy value of the i-th intelligence.

10.3.4 Real-Time-Coordination

During resource allocation, intelligences should be able to share present information and adjust the demand and supply of resources respectively. The steps can be described as the co-ordination of an inefficiency allocation given the underlying ineffortability, leading to a dynamic process where if, for example; supply and demand of a resource shifts or when individual insufficiency changes in individual intellegences' demand or supply arises in their perception there may arise need to re-coordinate the resouce allocation scheme. In order to deal with continually changing environment distributed computing method is employed to help with resource management and real-time adjustments.

10.3.5 Collective Learning and Optimisation

In such an event, through the incorporation of pattern recognition and reinforced learning, allocation strategies can be flexibly assigned relative to observed trends and emergent scenarios. This helps our intelligences to keep learning and evolve more of their decision making capabilitiesa leading to vesian, agile approach in terms of managing resources that adapts the changing dynamics of Mega-Bikes environment. The use of collective learning will also enhance resource allocation strategies thus increasing the potential score of the agents or length of survival.

10.4 Defense strategies

Team 6 group's defense strategy is an energetically flexible distance-based system, which gives an upper hand for the intelligences to survive and compete in SOMAS World. The presented approach combines two fundamental

elements, supplemented by predictive mechanisms as well as evading algorithms.

Firstly, Energy Management: The temperature level at a certain energy state should be home to Intelligences which are prepared to move. If an Intelligent Entity lasts long enough that the energy level becomes null and the current Mega-Bike is distant from desired Lootbox, it will detach itself from existing Mega-Bike to find any nearest available new to map Mega-Bike towards desired Lootbox. The company will thus restore its energy so it can be to the "normal" that precludes potential assaults from FWAwdiDriver or eliminations due to lack of energy. The use of machine learning algorithms to predict energy depletion and propose the best times for recharging boosts this aspect.

Controlling distance is essential for Intelligentsia. They must maintain a certain distance to escape or pursue. If an Intelligence is close to the FWAwdiDriver, it will prioritize escape to avoid capture or interference by a third Intelligence, regardless of other conditions. The Intelligent Body can make quick decisions on evasion by using advanced techniques and real-time spatial analysis. These decisions are based on the proximity of the threat and the structure of the environment. Further details will be provided in a subsequent section.

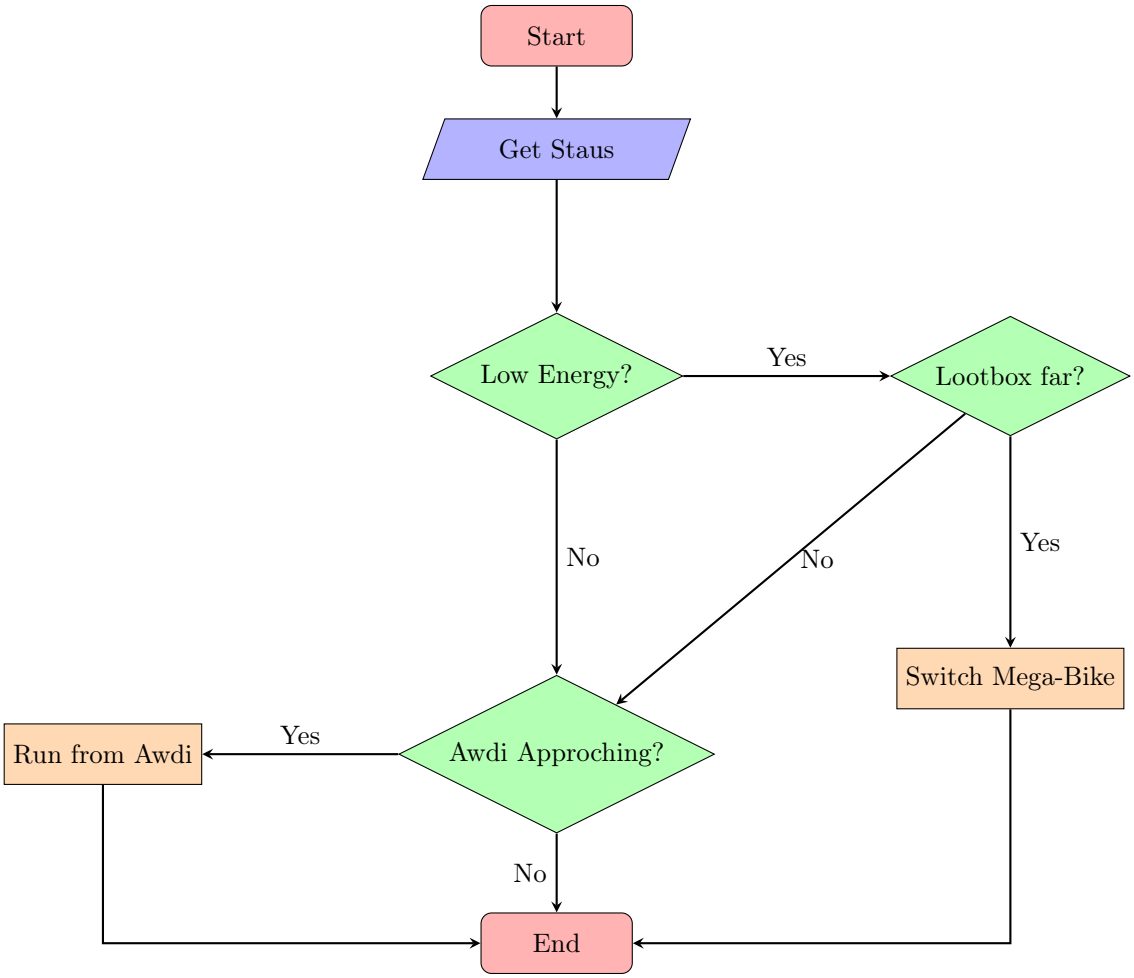


Figure 10.5: Defense strategies flowchart

10.4.1 Run from Awdi

"Courage is resistance to fear, mastery of fear, not absence of fear."
-Mark Twain

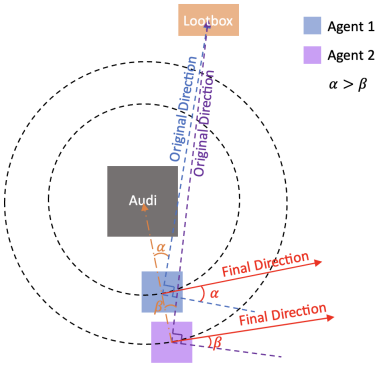


Figure 10.6: Escape scenario

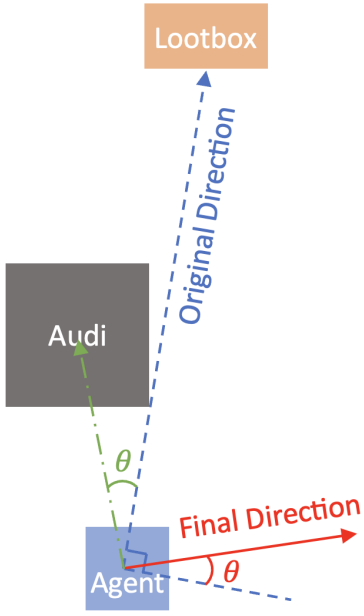


Figure 10.7: Turning Angle Calculation

During experimentation, we found that running in the opposite direction was very inefficient, as Awdi is very slow, so in many cases it would even be a pointless waste of energy, resulting in the death of the agent. Therefore, Team6's agent has developed a comprehensive evasion strategy that guarantees stable evasion of Awdi's pursuit while conserving maximum energy and staying on course towards the target Lootbox.

The deflection angle is determined by the angle between the ray connecting the two points from Mega-Bike to the target and the ray connecting the two points from Mega-Bike to Awdi. If the deflection angle is greater than 90 and less than 180, Agents will not need to perform a dodge reaction.

Algorithm 12: Escape Strategy

Data: Distance(x,Awdi),DeflectionAngle(x,Awdi)
Result: TurningAngle
foreach *agent(x)* **do**
  if $Distance(x, Awdi) \leq Threshold$ **then**
    if $DeflectionAngle(x, Awdi) < 180 \vee DeflectionAngle(x, Awdi) > 360$ **then**
      $TurningAngle \leftarrow GetAngle(x)$;
  else
    $TurningAngle \leftarrow 0$;

Refer to the analysis diagram above for more details. When Awdi approaches Mega-Bike within a certain distance, the agent selects the appropriate Turning Angel based on the distance and anglebetween them. If Awdi is far away, Team6's agent has enough time to change direction and evade. If Awdi is close, the agent has little time to react and evade the pursuit.

The figure shows the Turning Angle of two Agents, x and y, in relation to Awdi. Agent x is closer to Awdi and has a Turning Angle of α, while Agent y is farther away and has a Turning Angle of β. From this, we can derive: $\alpha > \beta$

Detailed test results can be found in the Results analysis section. Team6 collected scores that are significantly

higher than other teams.

10.5 Exploration strategy

10.5.1 Procedure

Intelligences must locate and retrieve Lootboxes in different colours to accumulate energy as well as scores in the Mega-Bike system. The Lootboxes consist of energy bars which restore the energizes. A colour target is assigned for each intelligence, and this colour scheme changes randomly every time a Lootbox of the new colours had been retrieved. Intelligences have to work together to formulate the rules and plans of target selection, succession strategy, resource allocation and admission of new members.

To enhance the efficiency of searching for Lootboxes, Team6 propose the following strategy:

The Intelligentsia will continuously search for and collect Lootboxes on the Mega-Bike

If the Intelligentsia fails to obtain the target-coloured Lootbox after eight consecutive attempts, they will dismount the bike. This allows the Intelligentsia to join other Mega-Bikes and attempt to collect new targets.

Dismount the bike when low on energy

An Intelligent Body's energy depletes over time. If the Intelligent Body's stamina falls below a certain value, it will dismount the bike. Dismounting allows the Intelligent Body to join other Mega-Bikes and replenish its energy.

Same-coloured Mega-Bikes should be prioritised when dismounting

An Intelligence will choose a Mega-Bike of same colour. Mega-Bikes of same color share similar goals and are thus, easy to collaborate with to the same course.

Consider Mega-Bike ratings

The Intelligence should dismount due to Mega-Bike rating. Mega-Bike has more superior performance and increased probability to get Lootboxes for Mega-Bikes of higher merit.

Intelligence will also take note of Mega-Bike's seat when getting off. If a seat is available, then the Intelligent Body will attempt to occupy it.

The method could make Lootbox search more effective and decrease risk of failure.

10.5.2 Evaluation

The search strategy comprises three main aspects: goal-directedness, energy constraint, and environment awareness.

- The ability of the intelligence to exhibit goal-directedness manifests in a tendency to prefer opening Lootboxes of similar colour with its target.
- Serial decisions relate to so-called low energy constraint that guides an agent against searching Lootboxes that are too far.
- Environment awareness implies that an intelligence adjusts its search strategy depending on the environment in which it is operating, including hostile intelligences.

Team 6 behaves as follows in the real SOMAS World:

Intelligence will estimate a score for each following Lootbox search. If the vehicle group does not score any point in eight successive Lootbox searches, then we qualify its current search strategy as "no-effect". After that, the intelligence will dismount from the vehicle and rate this as negative for a group of vehicle.

If a certain energy threshold is met, it means that the current team's search strategy does not provide enough energy for the intelligence. The intelligence will therefore leave the team and rate them negatively.

The intelligence will sort all Lootboxes by distance and colour using a function.Based on its target colour and energy, the intelligence will choose the appropriate vehicle group.

Intelligences with higher stamina will readily go for a Lootbox of the same colour to their intelligence. Otherwise, they will choose the vehicle group with the highest-rating among all vehicle groups.

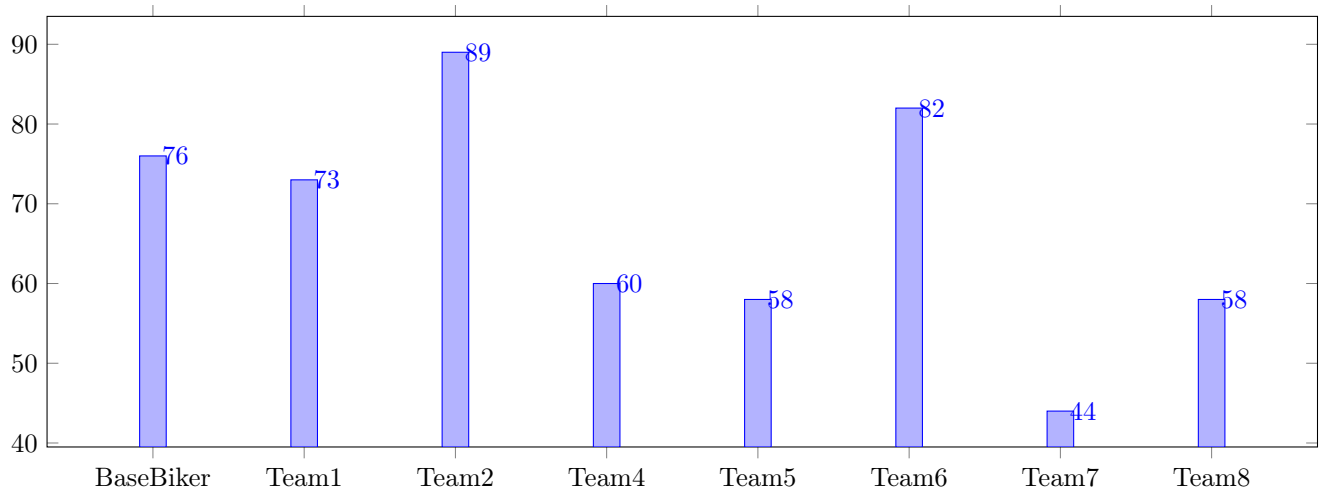
Lower-stamina intelligences will also opt for the topmost rated vehicle group. If the highest rated vehicle group does not exist, then any group with a rating higher than 0 will be considered.

The intelligence also considers the seating capacity of a vehicle group when selecting. The intelligence will try to join the vehicle group if it is not full and it will keep on looking for another possible group if it is full.

Team 6's search strategy has yielded positive results in experiments, with intelligences obtaining scores significantly higher than those obtained through random search strategies.

10.6 Conjectural Proof

Average Life Time per agent



The results chart shows that Team 6 has achieved remarkable success in the SOMAS World experiment. Our performance, with an average lifetime of 82 loops per agent, not only exceeds the baseline but also demonstrates our team's strategic acumen. It is important to maintain objectivity in our evaluation of the results.
Reflection of our effective collaboration and astute decision-making, as well as our agents' adeptness at navigating the complexities of the environment. We implemented advanced algorithms and adaptive strategies that enabled our agents to thrive where others faced challenges, demonstrating our innovative approach. The simulation showed that the approach was effective and successful.

How Forgiveness influence overall Agents survival

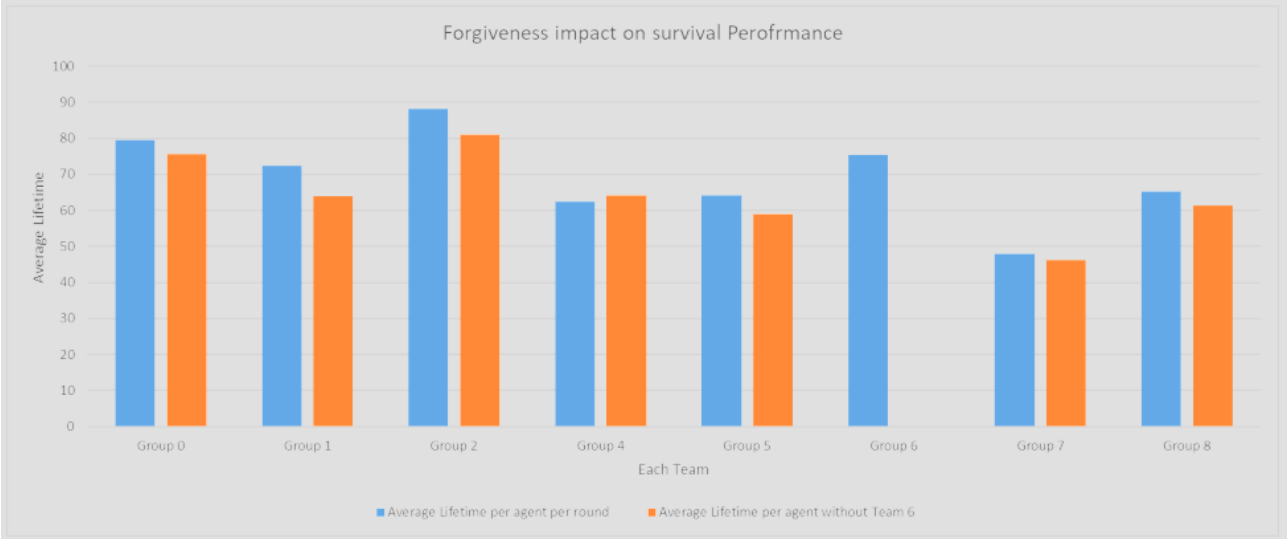


Figure 10.8: Comparison of survival with forgiveness or not

The key reason Team 6 offers a marked improvement in survival performance across teams, as shown by the blue bars is because of our innovative "forgiveness" strategy. This model helps to strengthen not only our own

agents' outcome but also the overall resilience in interactions and decision making of the SOMAS society. This strategy is effective; the contrast between the blue and orange bars in the chart shows that forgiveness as a legitimate element of our agents' strategic repertoire translates into more helpful attitudes, therefore, more successful community.

10.6.1 Defence strategy in real situation

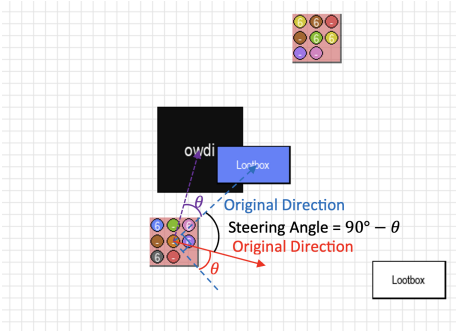


Figure 10.9: Escape scenario

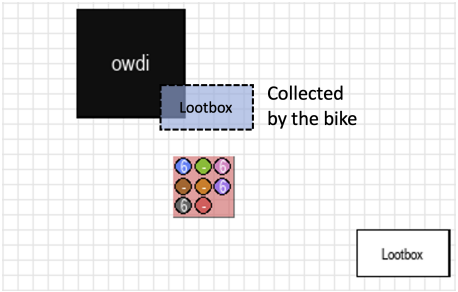


Figure 10.10: Collect the Lootbox and escape successfully

From the visualisation map above , it can be seen. When Awdi is chasing the Team 6 agents, the intelligences will not just dodge in the opposite direction, but will design the optimal route to save most of their energy while ensuring safety.

The figure on the left shows the scene when Awdi is chasing Team 6 agents, and the picture on the right shows that intelligences have successfully evaded Awdi and collected the Lootbox.

10.7 Future work

Although the overall performance of Team 6's agents is satisfactory, improvements can still be worked on in the future to enhance the survival time and scores. The improvements can be divided into two key aspects: Governance and Messaging.

10.7.1 Governance Improvements

Based on the strategy stated in the governance section, Team 6's agents prefer to choose Dictatorship. The agents might still fall into a situation where most of the agents on the field do not have a target colour that matches their own - Deliberative Democracy. Or, the Mega-Bike is in Leadership or Dictator of another target colour that differs from Team 6's agents. In these cases, according to the experiment results, the agent should leave and change the bike as soon as possible to achieve a higher score and longer survival time.

10.7.2 Messaging Improvements

The current messaging system must be more comprehensive and cannot contain complicated messaging functionalities. The overall performance could be improved by completing the handle joining message function. As a result, more detailed communications can be made with other intelligences on the same field (Mega-Bike), especially those with the same target colour. They can communicate and select a dictator with common interests to maximise overall benefits. They can also communicate with the same target intelligences before leaving the bike to update their reputation and choose another Mega-Bike to join together. Doing so might result in a Mega-Bike that only consists of intelligences with the same target colour so that the score might be optimised.

Chapter 11: Team 7

11.1 Introduction

Meet Agent 007, Team 7's personality-defined agent. Agent 007 has defined personality traits that affect their trust and opinion of other agents, which then in turn dictate different behaviors in SOMAS World when required to cast votes or make decisions. In the following sections, we will examine the idea of personality and how we implemented this within our code. Next, we will explain how we form opinions and a social network based on trust. Then, we will show how we use personality, our social network, and opinions to make decisions and cast votes within SOMAS World. Finally, we will show various ways that we have improved our agent based on initial experiment results.

11.2 Personality

11.2.1 Motivation: Why Personality?

Personality has been integrated into Multi-Agent Systems (MAS) for a variety of reasons. Firstly, it provides a means of mimicking natural intelligence, particularly in simulations which aim to model human behaviour such as emergency evacuation simulations [13], [4]. Thus, giving Agent 007 a personality allows us to mimic natural intelligence, seeing how different human traits influence performance within SOMAS World. Secondly, it enhances the interpretability of simulation results [13]. This appealed to our team as Agent 007's performance in simulations could be attributed to its personality. Then, Agent 007's personality could be altered to see how different personalities do in SOMAS World; Agent 007's behaviour, and therefore performance, could be changed by simply modifying its personality traits.

11.2.2 Theory: Five Factor Model

Figure 11.1 shows the Five Factor Model, also known as OCEAN, which distills personality into five traits [40]:

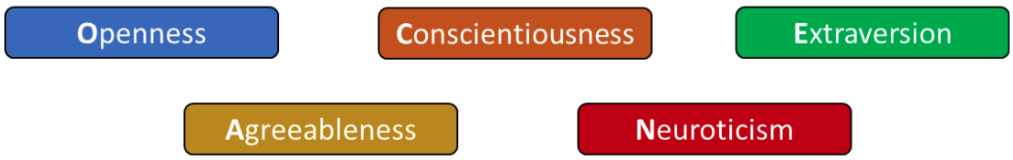


Figure 11.1: OCEAN Model Personality Traits

OCEAN is a widely accepted model for personality in fields such as social psychology as it provides a succinct yet thorough framework for evaluating one's personality [3]. It has also been incorporated into agents along with emotion frameworks to create an extended version of the Beliefs Desires Intentions (BDI) agent architecture [2]. The interpretations of high and low values of each trait are shown in Figure 11.2.

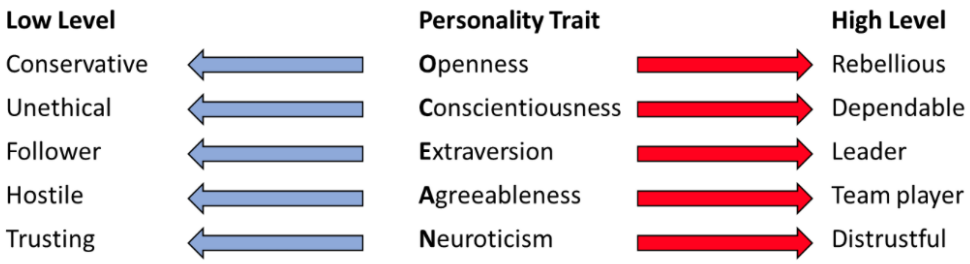


Figure 11.2: Interpretation of Five Factor Model Traits

11.2.3 Implementation

We provided Agent 007 with four of the five traits in the OCEAN model and assigned a value between 0 and 1 to each personality trait, with 0 representing a low level of the trait and 1 representing a high level. For example, an agent with a high level of extroversion was outgoing and more likely to send messages to other agents. We interpreted an agent with a high level of conscientiousness to be diligent and pedal with effort equal to its energy level. High agreeableness resulted in a cooperative agent, less likely to lie in messages and more likely to share its votes for allocation and Lootbox proposal with others. Finally, a highly neurotic agent was biased towards being distrusting of other agents. Throughout the rest of this chapter, we will present the various ways that we used these personality traits to influence Agent 007's behaviour.

11.2.4 Future Work

In future work, Agent 007 could use openness – the first personality trait of the OCEAN model – in decisions such as voting to accept a new agent requesting to join its bike. Furthermore, Agent 007 currently only uses one trait at a time to influence different actions; future work would incorporate multiple different personality traits into each action it makes.

Moreover, due to time constraints, Agent 007's personality did not evolve during the simulation. Future work would include implementing a Genetic algorithm to optimise the personality profile of our population of agents between rounds. A Genetic algorithm is an optimisation method inspired by natural selection and has been employed to tune the personality of agents in video-games [32]. In our case, the personality traits of elite agents would have been crossed-over and mutated to create offspring agents for the next round and the fitness function would have been the points accumulated by an agent in a given round.

11.3 Social Network

11.3.1 Motivation

The concept of the social network is rooted in the notion of memory, wherein agents recall their interactions with others, cataloging their experiences with each agent. This repository of memories enables the agent to make more informed predictions and decisions, incorporating insights from past interactions with other agents into their strategic considerations.

11.3.2 Theory

Social capital can be looked at through two different lenses: Ostrom and Ahn's focus on institutional knowledge and trust relationships [52], and Bourdieu and Wacquant's examination of social capital [8].

Our approach based on Ostrom and Ahn's theories centers on the knowledge of institutions and the strength of trust-based connections. They posit that a deep understanding of how institutions operate, coupled with robust, trust-based relationships within these frameworks, constitutes a vital form of social capital [52]. This understanding allows us to use our trust of other agent to make the best decision for our own agent as well as the collective. Through looking at these relationships in the context of Ostrom and Ahn, the trust relationships are a means of using our social network to make the decisions which will be of most benefit to our agent [52].

In the lens of Bourdieu and Wacquant, on the other hand, we can treat trust as a form of dynamic social capital. They suggest that social capital can be leveraged by those in institutionalised relationships for their own benefit, even without the knowledge of the individual who possesses this capital [8]. In Bourdieu's framework, social capital is one of several forms of capital (alongside economic, cultural, and symbolic capital) that individuals and groups use to maintain or enhance their position in the social hierarchy [8]. Trust, in this context, becomes a resource that can be used to gain favour from other agents. This favour can come in the form of being allocated resources or being voted in to a leadership position. However, the higher our trust relationship with another agent, the more likely we are to make decisions which benefit the other agent as well – in this instance, the trust relationship can be thought of as a measure of how much another agent may leverage our agent.

11.3.3 Implementation

Trust

Trust is formulated and updated by a socioeconomic and sociocognitive framework. This framework integrates the three dimensions of trust, i.e. cognitive, economic and normative. Trust is gained and lost based on a number of factors which include how agents vote on matters in the previous round as well as their choices in pedalling, braking, and steering.

Cognitive dimension: This aspect of trust involves the interpretation of signals, and knowing true actions from those which have been assumed. In the context of the agent, the values that are of highest priorities are:

- How hard each other agent is pedalling
- Which Lootbox direction each other agent has proposed

Per the guidelines, Agent 007 was only able to obtain these values from messaging, rather than directly calculating them. Thus, these values are not guaranteed to be true. This adds a layer of uncertainty to our trust calculation; other agents may lie in their messages. Our current implementation of Agent 007 assumes that the values gotten from messaging are honestly reported and therefore correct. This may, however, inflate our trust levels higher than they should actually be.

Normative dimension: This dimension incorporates trust factors related to the agent's personality, specifically influencing the rate at which trust is gained or lost. The following personality traits influence this rate:

- Neuroticism: The more neurotic an agent is, the less trusting and more skeptical it is. Therefore, when interpreting how each other agent reported that they voted on resource allocation, a small allocation for Agent 007 would lead to more trust being lost if Agent 007 is highly neurotic, and less trust being lost if Agent 007 is not very neurotic.
- Agreeableness: The more agreeable an agent is, the more willing it is to be a team player; in Agent 007's case, this means that it is more able to overlook receiving small resource allocations, attributing these small allocations to the need for teamwork. Thus, the more agreeable Agent 007 is, the less trust will be lost after another agent reports having given Agent 007 a small resource allocation.
- Conscientiousness: The more conscientious an agent is, the more dependable and ethical it is. The more conscientious it is, the more it would expect others to also be ethical and dependable. Therefore, a highly conscientious Agent 007 will be more distrustful of another agent if they pedal less.

Forgiveness plays a vital role in the dynamics of trust, particularly in the context of interpersonal relationships and community interactions. When trust is compromised, forgiveness becomes a critical mechanism for healing that trust. Here it is implemented in Agent 007 as the opposite of losing trust [19].

Economic dimension: This dimension is concerned with the level of risk being taken as some decisions do not require a high level of trust of other agents. The level of risk is then weighed with the possible reward of taking the risk.

Social Network

The social network of Agent 007 is made up of individual connections, each with an agent that is currently on the same Mega-Bike as Agent 007, or with an agent that was previously on the same Mega-Bike as Agent 007. Each connection consists of the duration of the connection, recorded as the length of time spent on the same bike with the agent at the endpoint of the connection, and a measure of trust of that agent (a normalised value between 0 and 1).

Figure 11.3 displays a visualisation of Agent 007's social network, consisting of all agents currently on their bike, as well as a few other agents that had been on it previously.

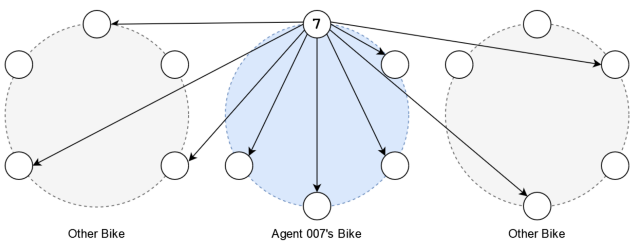


Figure 11.3: Social Network Visualisation

11.3.4 Future Work

The trust calculation framework could be enhanced by incorporating more ways of calculating "fairness" and expected values. There is also potential to explore more intuitive methods for calculating unknown values. Developing algorithms or models that can more effectively infer or approximate these values, such as employing a

genetic algorithm, could significantly improve the robustness and reliability of the trust calculation process. Furthermore, other factors of personality can be incorporated into the trust calculation mechanisms. By addressing these areas, future iterations of the trust framework could be used in making better decisions.

11.4 Opinion Formation

11.4.1 Motivation

Our opinion framework allows Agent 007 to make informed decisions by integrating diverse information sources, aligning beliefs and preferences, and guiding actions based on calculated opinions.

11.4.2 Theory

Our opinion framework is based on social influence theory, particularly drawing from network theory and cognitive psychology. It integrates how individuals are influenced by their social networks and how their opinions evolve through interactions.

11.4.3 Implementation

Information sources: Decides which information is available to the agents from messaging, enabling diverse inputs that impact opinion formation.

Agents' Beliefs and Preferences: This aspect shapes the priorities and choices of the agents, influencing the weight assigned to different information sources.

Integration with Action: Links the opinion framework with the agents' actions, ensuring that opinions inform and guide decision-making or messaging processes.

Messaging Dynamics: The use of messaging in our agents' opinion formation played a pivotal role, as it shaped their awareness of the environment and the actions of other agents, laying the groundwork for opinion development. Messaging served as a conduit for disseminating information among the agents, while also determining the availability of information sources. It was instrumental in establishing the agents' beliefs and preferences, sharing their individual priorities and choices, thereby aiding the collective in understanding and influencing the decision-making process.

Through messaging, the opinion framework was connected to the actions of other agents. This interplay meant that the opinions formed through messaging directly influenced subsequent decisions related to pedalling, braking, or steering in the SOMAS World.

The Ramirez-Cano-Pitt model was integral in guiding our understanding and implementation of voting and decision-making processes. Messaging facilitated communication and negotiation within this framework, significantly affecting governance choices, such as the adoption of deliberative democracy to establish agents with greater influence power (e.g., leadership democracy or dictatorship).

Additionally, we leveraged messaging to cultivate trust among agents, both from the same bike and those encountered previously. This capability enabled agents to comprehend and anticipate the behaviors of others, proving crucial for opinion formation and its linkage to strategic actions.

To calculate opinion, we used the following equations from the **Ramirez Cano Pitt Model**:

$$o_i\left(t'\right) = \sum_{j}^{j \in SN_i} w_{i,j}(t) o_{i,j}(t) \tag{11.1}$$

$$w_{i,j}\left(t'\right) = \frac{w_{i,j}(t) + w_{i,j}(t) a_{i,j}(t)}{\sum_{k}^{k \in SN_i} \left(w_{i,k}(t) + w_{i,k}(t) a_{i,k}(t)\right)} \tag{11.2}$$

$$a_{i,j}\left(t'\right) = 1 - \frac{\left|o_{i,j}(t) - \mu_i\right|}{\max\left(\mu_i, 1 - \mu_i\right)} \tag{11.3}$$

Equation 11.1 calculates the updated opinion of agent i at time t' by aggregating opinions of neighboring agents j weighted by their influence $w_{i,j}(t)$.

Equation 11.2 determines the updated influence weight between agents i and j at time t' based on their previous weight $w_{i,j}(t)$ and an adjustment factor $a_{i,j}(t)$.

Equation 11.3 calculates the adjustment factor $a_{i,j}$ at time t' by considering the discrepancy between the opinion of agent i and the mean opinion in the network.

11.4.4 Future Work

Future iterations of our framework aim to incorporate temporal dynamics in opinion formation models, consider adaptive learning mechanisms, and explore real-world applications in dynamic environments.

11.5 Voting and Decision-Making Processes

11.5.1 Governance

Theory

The adoption of Deliberative Democracy in our system is underpinned by the Condorcet Jury Theorem [7], which asserts that larger groups of independently deciding agents are more likely to reach accurate collective decisions. This characteristic positions Deliberative Democracy as a robust choice for collective action within multi-agent systems. Its inherent adaptability to complex problems is a key strength, facilitating a balance between the varied perspectives of diverse agents and the need for specialized expertise. From a historical viewpoint, this model boasts a proven track record of success, exemplified by its effective implementation in ancient Athenian governance. Such historical precedent not only underscores its potential for fostering long-term stability and cooperation but also validates its application in contemporary multi-agent settings. While recognizing its potential limitations in scenarios that demand rapid, high-stakes decision-making, our implementation of Deliberative Democracy has been comprehensive, laying a solid foundation. This approach leaves room for future integration of other governance models, such as leadership democracy or dictatorship, which may offer alternative solutions in fast-paced or high-threat contexts.

Implementation

Initially, agents engage in preliminary, non-binding votes, allowing them to freely express preferences and propose ideas without immediate consequences. This phase fosters open dialogue and helps in understanding the collective stance of the group. Following this, agents participate in final binding votes, where decisions are solidified based on majority consensus. This method ensures that decisions are made democratically, reflecting both individual preferences and collective wisdom.

Future Work

While deliberative democracy offers a robust framework for collective decision-making in our system, excelling in scenarios that benefit from diverse perspectives and require a balance between individual and collective goals, its application in high-risk, fast-paced scenarios necessitates careful consideration. The potential inefficiencies and risks associated with delayed decision-making in such situations must be acknowledged. To address these limitations and enhance system functionality in a broader range of scenarios, future work will explore the implementation of alternative governance models such as leadership democracy or dictatorship. These models could provide more streamlined decision-making processes and rapid action capabilities, thereby complementing the existing deliberative democracy framework.

11.5.2 Navigation

Theory

The decisions on how navigation functioned in our system were derived from Axelrod's Prisoner's Dilemma Tournament [5]. This explores how cooperation and competition influence decision-making, particularly in scenarios where individuals must choose between personal benefit and collective good. This concept is extremely relevant to the pedalling dynamics in SOMAS World, as each agent must decide whether the best option is to pedal, thereby contributing to the collective movement of the Mega-Bike, or to conserve energy for personal benefit. This mirrors the cooperate-defect choices presented in the Prisoner's Dilemma.

Implementation

Based on the theory, we implemented a pedalling strategy where the amount of pedalling input from each agent was directly proportional to their current energy levels. The ratio between the agent's pedalling force and its energy level was set to the conscientiousness level (C) of the agent. Therefore, an agent with a high conscientiousness level would be more diligent and hence, pedal with a high force (F) relative to its energy

level (E). This aimed to balance individual energy conservation with the collective need to keep the mega-bike moving forward. Consequently, our team initially chose not to implement a specific braking strategy to prevent potential exploitation by other agents. However, considering challenges in SOMAS world, such as the Awdi threat, we plan to include braking in future strategies for collision avoidance and safety enhancement.

$$F = C \times E \tag{11.4}$$

Our steering strategy primarily aimed at optimizing navigation towards specific targeted loot boxes. Thus, the steering direction was calculated based on the selected loot box, facilitating a coordinated approach to achieving specific objectives. This implementation draws from the theoretical aspects of agent decision-making in a resource-constrained and goal-oriented environment.

Future Work

In future work, we would refine the pedalling strategy to be based entirely on expected utility. This would involve a more nuanced approach, allowing agents to assess the potential benefits and costs more effectively, thereby enhancing decision-making efficiency in resource allocation. Furthermore, we would incorporate personality traits such as conscientiousness to control the degree to which we pedalled or steered in the correct direction, with higher conscientiousness leading to pedalling more and steering the Mega-Bike-decided direction, and lower conscientiousness leading to pedalling less or steering erratically. Furthermore, considering challenges in SOMAS World such as the Awdi threat, we could include braking in future strategies for collision avoidance and safety enhancement.

11.5.3 Direction Proposals

Theory

For the first stage of democratic voting, Agent 007 must propose a Lootbox. When making a proposal, it is better to make a selfish proposal to ensure that it is possible for our desired outcome to be considered in a final vote. For the second stage of democratic voting, Agent 007 must vote on proposed Lootboxes. When voting, it may be advantageous to consider others' desired outcomes, depending on what our opinion is of each Lootbox proposal, as well as how agreeable we are.

Implementation

For the first stage of democratic voting, Agent 007 must propose a Lootbox. Our implementation is simple and selfish: if we are below a certain energy level (0.25), then we propose the nearest Lootbox. If we are above that energy level, then we propose the nearest Lootbox of our colour.

For the second stage of democratic voting, Agent 007 must vote on all proposed Lootboxes. Our agent first considers our opinion of each proposed Lootbox; if our opinion is above a certain threshold (0.5), then we consider it a viable candidate (our own proposal is automatically considered a viable candidate). Next, our agent considers our agreeableness level from our personality: when high, we are more of a team player, when low, we are more selfish. Thus, the higher that our agreeableness is, the more equally we distribute our vote among viable candidate Lootboxes. The lower that our agreeableness is, we hoard more of the vote for our own proposal, then distribute the rest of our vote among other viable candidate Lootboxes equally.

Future Work

Future work in this area could include more strategy incorporated in the initial proposal stage of voting; Agent 007 could incorporate our opinion of Lootboxes to consider whether another Lootbox might be a more advantageous choice that would encourage cohesion amongst the group, encouraging everyone to pedal hard and get to desired Lootboxes. Furthermore, we would create more specific strategies for non-democracy governances; for example, we would incorporate personality traits and opinion to alter the dictated direction we choose when dictator; the more "agreeable" of a dictator we are, the more likely we would be to not only dictate our selfishly-desired Lootbox to the bike.

11.5.4 Other Agents

Theory

In order to make decisions about other agents, we primarily use our Social Network. We have formed a trust level for other agents throughout the games, which could be agents on the same bike as our own agent in the

context of the SOMAS World. Based on this trust/credibility, we can make logical decisions regarding the formation and structure of the Mega-Bikes we are on.

Implementation

The types of decisions we have to make regarding other agents include voting to kick agents off of our bike, voting to accept new agents onto our bike, and voting on a leader/dictator for our bike.

Voting to kick: When deciding whether to kick an agent off of our bike, we calculate our average trust of an agent over previous iterations. If that average trust is less than a certain threshold (0.4), then we vote to kick them off. If we don't have any trust history for them, then we vote to keep them.

Voting to accept: When deciding whether to accept an agent onto our bike, we do the opposite of voting to kick: if our average trust of that agent over previous iterations is over a certain threshold (0.4), then we vote to accept them. If we don't have any trust history for that agent, then we vote to accept them.

Voting for a leader or dictator: When voting for a leader or dictator, we are expected to return a normalised distribution of votes. We determine this distribution based on their average trust levels; they get an amount of our distribution that is directly proportional to our average trust levels of them over previous iterations. If we don't have any trust levels for them, then we give them a placeholder trust level of 0.5.

Future Work

Future work for this section would include incorporating our opinion of each agent in our decisions about them. This opinion would then be moderated by a personality trait such as agreeableness, which would decide to what degree we use that opinion to make decisions versus sticking with our own belief about trustworthiness.

11.5.5 Allocation

Theory

When making decisions about how resources from an acquired Lootbox should be distributed, we consider how agreeable we are (from our personality). This allows us to vary how much we want to consider others' needs and wants, depending on how agreeable we are.

Implementation

To determine a normalised distribution of votes, we divide up a portion of the vote equally among other agents on the bike that is directly proportional to how agreeable we are. For example, if we have an agreeableness of 1, then we give everyone on the bike an equal share of our allocation vote. If we have an agreeableness of 0, then we give Agent 007 100% of our vote and all other agents no allocation votes. If we have an agreeableness of 0.5, then we divide up half of the vote equally among all other agents, then give Agent 007 the remainder of the vote.

Future Work

Future work for allocation voting would be to incorporate opinion in our decision-making. As we did with voting for Lootbox proposals, we could determine which agents we even consider sharing our allocation vote with based on our opinion of each agent, then use our agreeableness to determine to what degree we share with them.

11.5.6 Outgoing Messages

Theory

We incorporated the agent's extroversion and agreeableness personality traits into its decisions regarding when to send messages to other agents and when to lie in those messages. As previously mentioned in Subsection 11.5.2, it has been shown that in the long-term it is better for the agent to be honest and cooperative with other agents. Therefore, our default agent had high agreeableness to prevent it from lying.

Implementation

Our agent was designed such that high extroversion would make it more probable to send messages and interact with other agents. To decide whether the agent would send messages, at each iteration of the game we generated a random number between 0 and 1. As shown in Figure 11.4, if the agent's extroversion parameter was greater

than the random number, it sent messages. Otherwise, the agent did not send messages. Similarly, if our agent had a high level of agreeableness, it was less likely to lie in messages. Hence, at each iteration another random number between 0 and 1 was generated. If the agent's agreeableness level was below this number, it lied in messages. For instance, when lying about force, the agent claimed it had pedalled with maximum effort to appear to other agents as cooperative. Furthermore, when it lied about its vote to kick another agent off its bike, it claimed to had voted "no" to avoid retaliation from the agent.

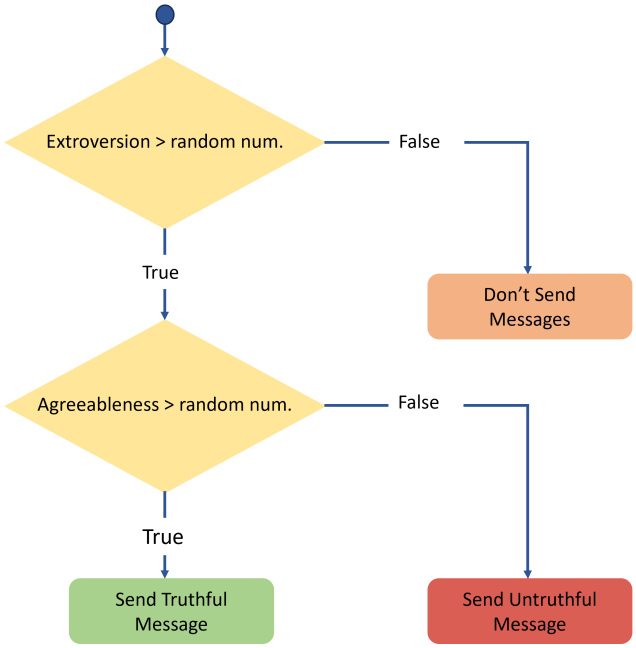


Figure 11.4: Team 7 agent decision flowchart for outgoing messages.

Future Work

In future work the agent's personality could be used not only to determine when it lied, but also the degree to which it lied. Many of the messages are more complex than simple yes/no binary answers, so, for example, we could say that we only pedalled a little more than we actually had, or we could lie and say that we pedalled with full force when we hadn't pedalled at all. The degree of this lie would be proportional to a personality factor such as conscientiousness.

11.6 Agent 007 Improved Performance:

In the initial experiments, Agent 007 exhibited suboptimal performance, consistently ranking last in several tests. In response to this, modifications were made to its personality parameters to enhance performance. Notably, Agent 08 adopted a 'free-rider' approach. Despite Agent 08's lack of pedaling, Agent 007 continued to trust it, leading other agents to abandon the bike and forcing Agent 007 to exert all the effort, indirectly supporting Agent 008's survival. As discussed in the Agent 007 Trust Section, this trust was established based on criteria that did not consider the pedaling input. The decision not to calculate pedaling input through the energy decrease of other agents over an iteration was based on the rationale that such a metric would not align with real-world scenarios. Consequently, if an agent chose the same Lootbox as us or if Agent 007 obtained its preferred Lootbox sufficiently often, it would continue to participate. However, this strategy led to a rapid depletion of Agent 007's energy, resulting in premature termination and a consequent loss of points over the course of 100 iterations.

To counteract this, the threshold for leaving a bike was lowered, which initially resulted in prolonged periods in the void due to hesitancy in joining new groups. This change led to constant energy loss and subsequent termination of the agent. Further parameter adjustments, such as synchronizing steering behaviours with higher-performing agents and personality adjustments, fostered better alliances and significantly improved point statistics, as shown in Figure 11.5.

Had there been sufficient time, further experimentation with parameter adjustments might have yielded more significant improvements in both lifetime and points. The lack of notable enhancement in the lifetime after the

improvements were made is shown in Figure 11.9. It was concluded that the cause of this was that Agent 007's pedaling rate was consistently higher on average compared to others. This observation suggests potential areas for optimising energy management.

This highlighted the usefulness of the personality framework and the modularity of the Team 007 code which facilitated rapid strategic adaptations.

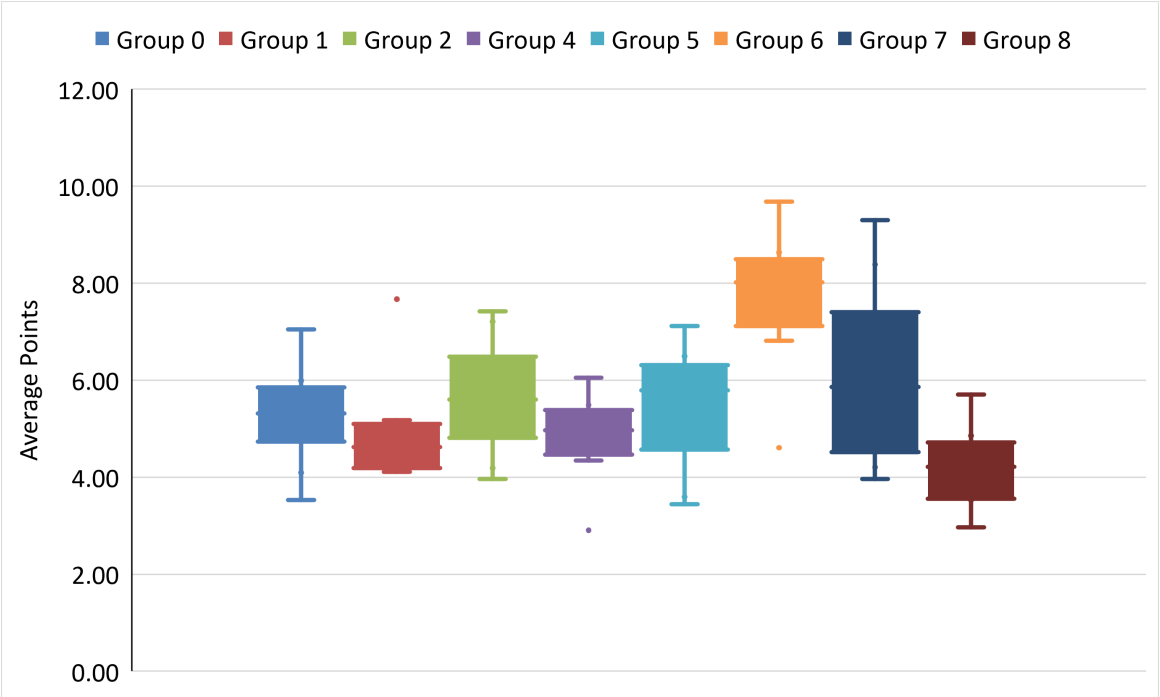


Figure 11.5: Average Number of Points, with Improvements Made to Agent 007

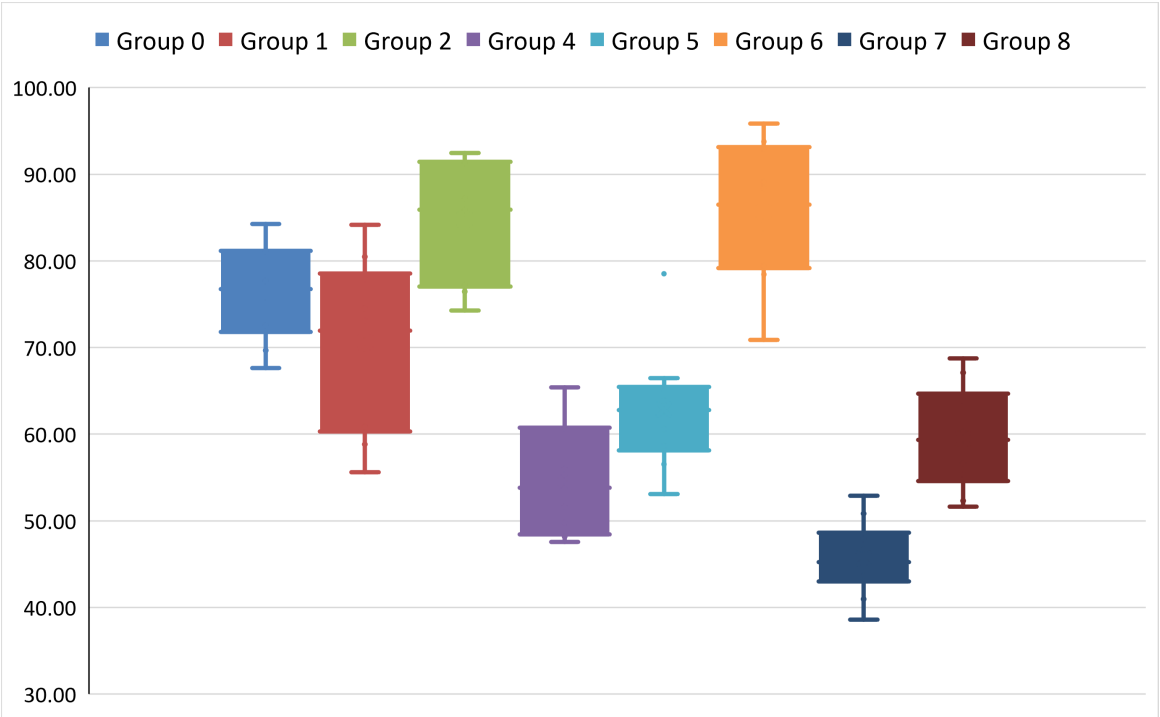


Figure 11.6: Average Lifetime, with Improvements Made to Agent 007

Leadership Cost Reduction:

Agent 007's performance was also improved with a reduction in the leadership penalty, consistent with its preference for democratic or semi-democratic leadership styles. However, the improvement was marginal. It simply meant that bikes leaning towards a dictatorial approach performed slightly worse, while Agent 007 fared slightly better. Given that the communication costs associated with full democracy were substantial, future implementations of this experiment could lead to finer adjustment of the cost to better represent the true cost.

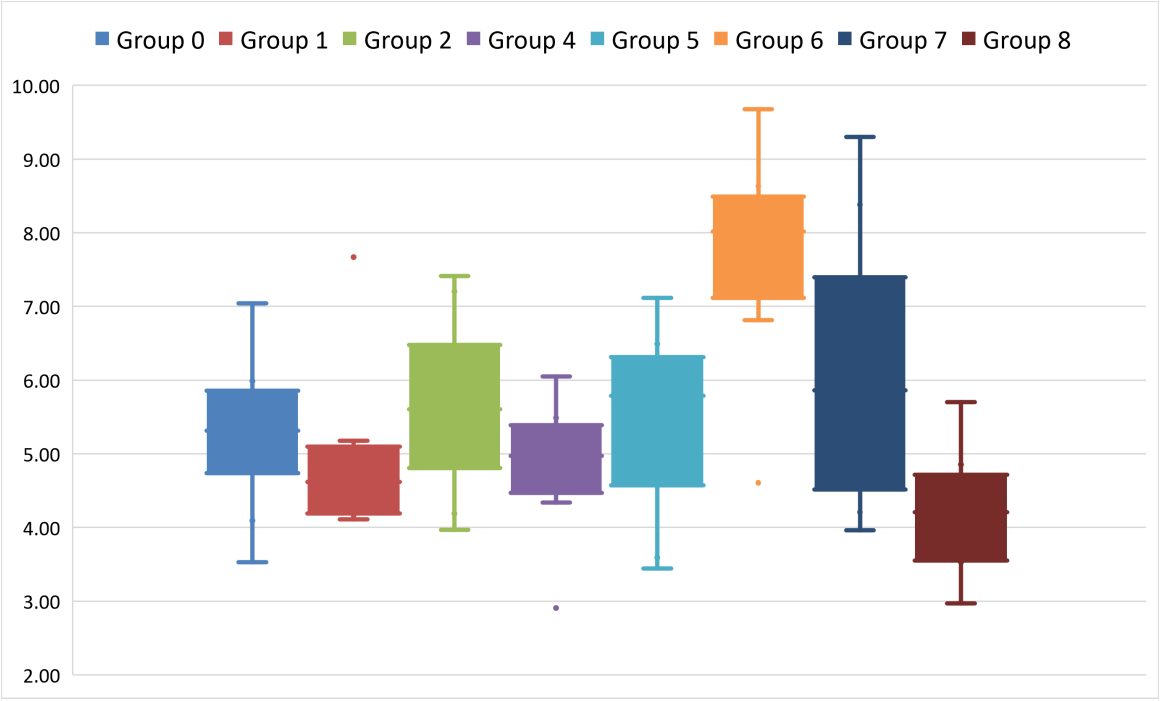


Figure 11.7: Average Points when Halving Leadership Cost

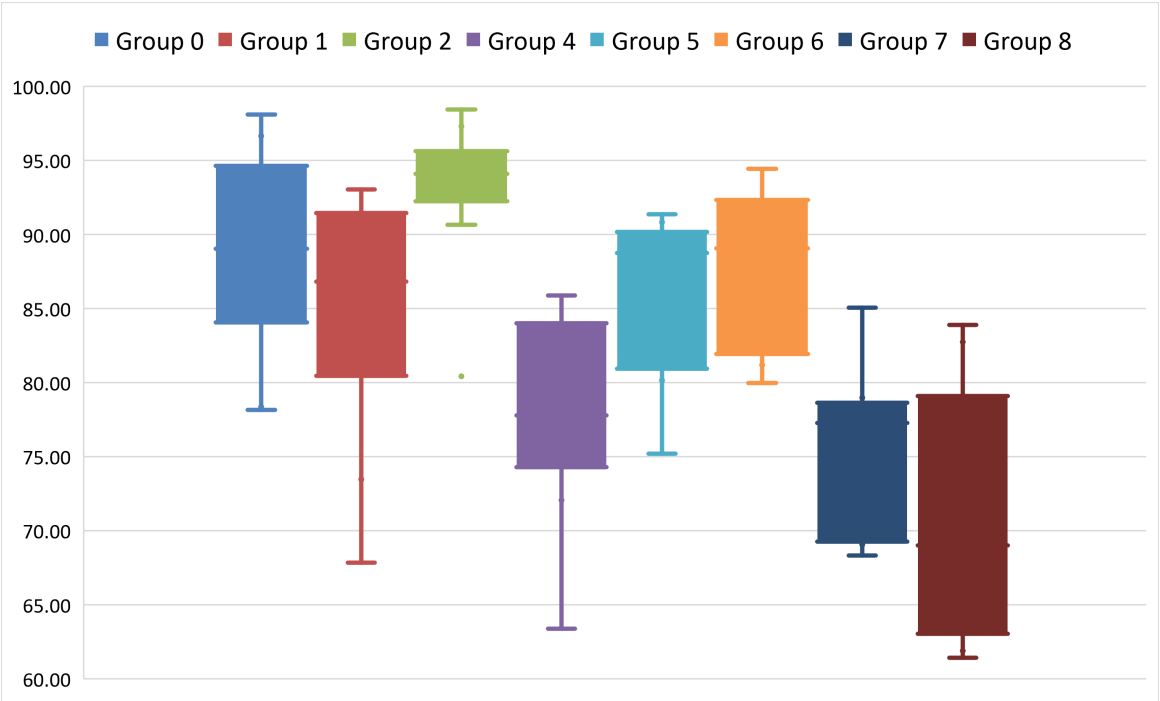


Figure 11.8: Average Lifetime, when Halving Leadership Cost

Further enhancements in our agent's performance could be realised through a more optimal selection of leadership style, coupled with refined decision-making regarding direction. Opting for a dictatorship government

structure and aiming to be the leader, our agent could effectively target areas with a high concentration of lootboxes and minimal bike competition. This approach ensures agents on our bike are satisfied with their points and energy, reducing the likelihood of them leaving. It also aims to mitigate the 'flocking behavior' as depicted in Figure 11.8, where multiple bikes converging on the same lootboxes often lead to Megabikes chasing disappearing resources, resulting in their death. Implementing this focused strategy could significantly improve our agent's performance.

Figure 11.9: Screenshot Demonstrating the Flocking Behaviour of the Megabikes

Chapter 12: Team 8 Agent

12.1 Overview

Team 8 introduces an agent with the self-modification strategy, which stands as the agent's principal strategy. This feature enables the agent to autonomously adjust its operational parameters based on a self analysis framework, ensuring an optimal balance between resource utilisation and the achievement of set goals. In alignment with the established open system framework of the game and the core requirements outlined in Ostrom's Institution Theory, our agents are designed to thoroughly embrace the central tenets of Evolutionary Economic Theory. They are designed to adeptly navigate the dynamic game environment and consistently adapt their decision-making processes. Our focus on self-modification aims develop a multi-agent system that excels in adaptability, efficiency, and maintains a fair and equitable environment for all participating agents.

12.2 Top-level Strategy

12.2.1 Problem Formation

Operating within Open Systems as defined by Hewitt [25], our agents are intended to carry out their designated activities as well as negotiate 'non-functional hazards' within their surroundings. Agents' ability to constantly interact with and adapt to their environment should be ensured by this design approach. When combined with the basic principles of Ostrom's Institutional Theory [48], this approach offers a thorough framework for sustainable resource management and governance. By combining these frameworks, we provide our agents the ability to operate in complexly interconnected scenarios, encouraging cooperation, adaptability[56], and efficient resource management, which are critical for long-term survival and success.

12.2.2 Fundamental Theory — Evolutionary Economic Theory

An essential component of our project's approach is evolutionary economic theory, which highlights the importance of inter-dependencies and feedback mechanisms by emphasising the non-equilibrium processes that internally modify the economy.[27] This point of view recognises that internal dynamics and exchanges continually reshape the economy, which is never static. The idea explains how economic systems are adaptive and complicated. In line with this, feedback is a key component of our agent system's architecture and serves as a means of learning and introspection.

12.2.3 Principal Strategy

Discuss of "Self"

Considering the winning criteria of the game (ensuring agent's energy is not depleted and achieving the highest score), our strategy is framed under an Analytic Framework for Self-Organization[17] which encompasses three aspects: self-awareness, reflection, and interoception.

- **Self-awareness:** Internal representation of agent itself

  Includes the agent's satisfaction and pre-judgment of others based on its own perspective.

- **Reflection:** Evaluation of the agents' performance

  Takes a more macroscopic view of the overall results of each game round, providing the agent with value.

- **Interoception:** Attention to system well-being

  Represents physical parameters (energy levels, scores, desirable target colour) that are essential for decision-making process.

By employing this self-modification framework, agents can adapt their strategies dynamically based on internal and external factors, thus enhancing their chances of winning the game while maintaining a sustainable and competitive presence in the open system.

Self-Modification Mechanism

The foundation of self-modification lies in the agent's acute self-awareness, a cognitive metacognition that allows it to comprehend its own decision-making processes. The agent possesses an introspective capacity, continuously monitoring its actions and their repercussions.

Upon executing a decision or participating in a voting process, the agent engages in a sophisticated reflection mechanism, assigning positive or negative valence to the experienced outcomes. Positive outcomes trigger a reinforcement mechanism, enhancing the agent's self-trust and fostering a propensity for altruistic behaviors.

The crux of self-modification lies in the adaptive adjustment of decision factors' weightings based on the outcomes and emotional responses derived from the reflection process. More precisely, positive reflections serve as a catalyst for a positive feedback loop, favoring those that contribute to successful and cooperative interactions.

The refined decision factors and adjusted weightings encapsulate a distilled understanding of the environment, incorporating the lessons learned from past experiences to inform future decisions.

Flow Diagram

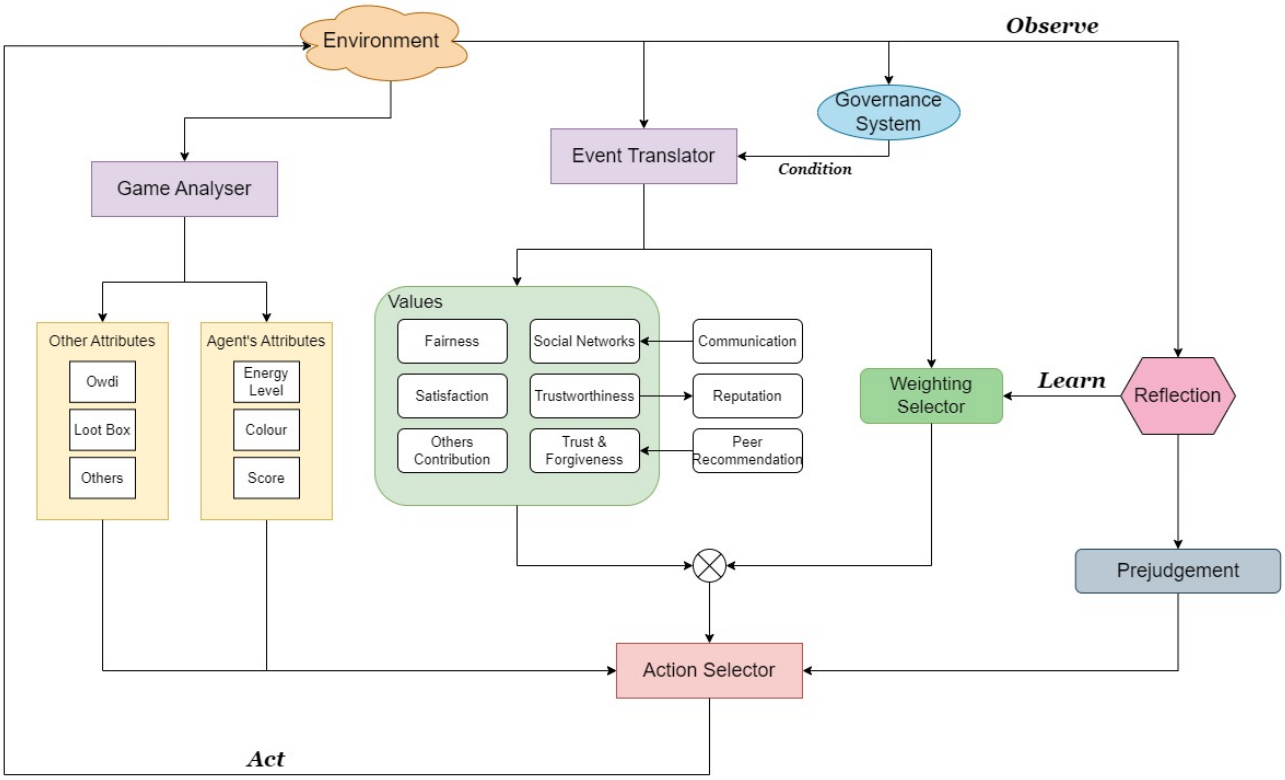


Figure 12.1: Team 8's Strategy Framework

Distilled from all supportive theories, Figure 12.1 illustrates the dynamic decision-making framework that continuously adjust weightings appended to agent's various evaluative dimensions to enhance overall performance.

The Game Analyser and Event Translator play pivotal roles in analysing the physical components of the game environment and comprehending specific tasks within each game stage.

There are two fundamental aspects for the game analyser: the agent's internal state, succinctly encapsulated as interoception, and the external attributes. The former serves as a reflective mirror of the agent's needs and individual goals, while the latter represents extrinsic factors, with a particular emphasis on the position and speed of Awdi, the color and distribution of Lootbox, among other essential considerations for survival.

From the aspects of the Event Translator, evaluation values are accessed, which encompasses the assessment of both self-state and overall performance, including fairness, satisfaction, others contribution, social networks, trustworthiness and trust&forgiveness. As the agent progresses through different game stages, it engages in continuous observation of each iteration's outcomes, thereby constructing a reflective process. When confronted with decision-making scenarios, the updated learnable weights are appended to evaluative dimensions. In addition, reflection extends to prejudge other agents' behaviour, as pre-requisite in game theory.

Furthermore, social networks are formed and solidified through communication, with trustworthiness and reputation closely intertwined. The factors of trust and forgiveness are thoughtfully considered peer recommendations.

12.2.4 Preference and Value

Energy level stands as the nucleus of all considerations, tightly interwoven with the actions of the agents. We conduct a comprehensive analysis of energy levels, considering it as a foundational element that permeates every decision-making process, forming the bedrock of our strategic framework.

Low Energy level

Our agent's behavior, influenced by Maslow's Hierarchy of Needs [39], is programmed to focus on basic survival needs when energy is low, reflecting Maslow's foundational level where physiological needs are critical. In low energy scenarios, the agent's goal is to secure essential resources for survival. This behavior aligns with the concept of Risk Aversion in Scarcity [18], where the agent shows increased risk aversion in resource-scarce situations, prioritising immediate resource acquisition over riskier, long-term benefits. In such conditions, minimising risk and maximising immediate utility becomes crucial for the agent's survival and continued functionality.

High Energy Level

In high energy states, our agents, with a surplus of resources, are inclined towards cooperative and altruistic behavior towards others in the system. Such a scenario aligns with the concept of Reciprocal Altruism. This principle refers to a cooperative mechanism where individuals extend benefits to others, anticipating similar favors in future interactions[73]. In such scenarios, the agent focuses on actions that promote social welfare, using its excess energy for the greater good of the system.

12.3 Design & Implementation

			Attributes			Institution	Values					Testing
Stage	Task		Energy Level	Colour	Score	Governance Type	Fairness	Satisfaction	Others Contribution	Trust & Forgiveness	Prejudgment	Previous Experiment
1	Vote Governance											☑
2	Stay or Leave											☑
	Which Mega-bike to Join											☑
3	Accept/Vote Off											☑
4	Lootbox Preferences	Non-binding										☑
		Binding										☑
5	Pedal and Steer Actions											☑
6	Vote Resource Allocation											☑

Strong Intermediate Weak N/A Dependent

Figure 12.2: Weightings for Values

The implementation phase focused on translating the agent design into functional code in Go. Agent functions were organized into modular Go packages, promoting code reusability and maintainability. Extensive testing was conducted to select the feature parameters wisely, e.g. energy threshold. The Figure 12.2 illustrates the initial weightings assigned to the decision-making driven values which are dynamically refined as game begins. The following section shows the coding procedure.

12.3.1 Reputation Score

The reputation score is a global variable that can be accessed throughout SOMAS World. We update it with a 'reputation' score based on the criterion of whether or not we have worked with agents in past rounds to perform organisationally beneficial behaviors (record in `ActionMap`). Forgivenss[41], as a mechanism to repair trust breakdowns, is used to scale the scores of past rounds of `ActionMap`, e.g. $lastroundscore * 0.9$, $secondlastroundscore * 0.9^2$ (as shown in Algorithm 13)

Algorithm 13: UpdateReputation Function for Agent8

```
Function UpdateReputation():
    initialise the agentCount and agentScore;
    for i = 1 to number of iterations we recorded do
        foreach agentId, Score in agentsActionsMap[i] do
            Sum up the agentScore using forgiveness ratio and increase agentCount;
    foreach agentId, scoreSum in agentScore do
        caclulate the reputation based on averageScore and messageReputation;
```

12.3.2 Self Reflection

The self-reflection, as core of the self-modification mechanism, is measured from three different angles.

1. At each iteration each collaborating agent is scored, reflecting the idea of trust&forgiveness, which is recorded in `agentActionMap` (as shown in Algorithm 14).

Algorithm 14: Function to update agent action map

```
Function updateAgentActionMap():
    currentLoopAgentActionMap ← new map;
    for agent ∈ all agents on current bike do
        currentLoopAgentActionMap[agent.GetID()] ← agent's energyLevel;
    for i ← 1 to number of loops to record - 1 do
        bb.agentsActionsMap[i] ← bb.agentsActionsMap[i+1];
    bb.agentsActionsMap[10] ← currentLoopAgentActionMap;
```

2. Score Mega-Bike at each iteration, monitor whether each Mega-Bike solves the coordination problem well and how well it performs in collective action, recorded as `loopScoreMap`. this score largely determines whether our agents want to switch and how they choose to do so (as shown in Algorithm 15).

Algorithm 15: Function to update loop score map

```
Function updateLoopScoreMap():
    calculate distance between bike and box for previous and current iteration and initialise loopScore;
    if don't get lootbox in current iteration then
        loopScore ← Score calculated using strategy based on distance change ration, energy loss;
    else
        loopScore ← ratio based on color of lootbox * score calculated based on energy gain;
    for i ← 1 to number of loops to record - 1 do
        bb.loopScoreMap[i] ← bb.loopScoreMap[i+1];
    bb.loopScoreMap[10][current bike] ← loopScore;
```

3. Individual satisfaction, based on resource allocation in relation to its demands (as shown in Equation 12.1). Demands are seen to be proportional to the efforts the agent made toward the organisational goals (explained in Reciprocal Altruism [73]). (as shown in Algorithm 16)

$$\sigma_{i,C}(t+1) = \begin{cases} \sigma_{i,C}(t) + \varphi \cdot (1 - \sigma_{i,C}(t)) & \text{if } r_i \geq d_i \\ \sigma_{i,C}(t) - \psi \cdot \sigma_{i,C}(t) & \text{otherwise} \end{cases} \tag{12.1}$$

Algorithm 16: Function to update satisfaction

```
Function UpdateSatisfaction():
    Initialise the parameter α and β;
    Calculate the distanceDiff from bike to target and energyDiff of previous and current iteration;
    if loss energy and get far away from target then
        bb.satisfaction ← value calculated based on alpha, previous satisfaction, satisfaction ratio;
        if target has the same color as what we need then
            bb.satisfaction is increased by small ratio;
    else
        bb.satisfaction ← lower value calculated based on previous satisfaction, beta;
```

12.3.3 Message System

An advanced communication system is implemented within the agent framework, utilizing Agent Communication Language (ACL) for the transmission and management of messages. This system encompasses two primary functions: `CreateMessage` (as shown in Algorithm 17) and `HandleMessage` (as shown in Algorithm 18).

`CreateMessage` function enables agents to construct and then send messages. These messages can cover a range of topics crucial for collaborative decision-making and strategy formulation. Key topics include:

- Proposals for kicking out bikers.
- Expressions of interest in joining the group.
- Preferences regarding Lootboxes.
- Determination of forces to apply.
- Decisions related to governance models.

Once receives messages, our agents employ `HandleMessage` function to process and interpret the information. The data extracted from these messages is integrated into the decision-making processes of the agents. It serves as a significant factor in shaping the actions and strategies of individual agents, influencing their responses to various situations within the simulation.

There is also a integral aspect of the communication system is the management of reputations through messages. Agents assess and update the reputations of their peers based on interactions and behaviors observed, which further informs their future decisions and interactions.

Algorithm 17: CreateMessage

```
Function CreateMessage():
    if we have low energy then
        Send kindly fake message and ask other agent to help us to survive;
    if we have lower score than others then
        Send message to ask other agents to help us gain more points;
    else
        Send 100% real message to gain trust from other agent;
```

Algorithm 18: HandleMessage

```
Function HandleMessage():
    Check the Message Type, the logic below is based on the Type;
    if message contains desired information then
        Add extra reputation to the sender;
    else
        if message contains selfish or harmful information then
            Reduce the reputation of the sender;
```

12.3.4 Choose Governance Systems

Our strategy (as shown in Algorithm 19) for governance is based on deliberative and leadership democracy, intentionally avoiding dictatorship. Deliberative democracy emphasises collective decision-making through discussion and consensus, allowing our agent to adapt to the game's dynamic environment through inclusive and diverse dialogue. Leadership democracy involves our agent actively seeking leadership roles, focusing on fairness and collaborative success. Dictatorship, avoided due to its restrictive nature, is contrasted with Arrow's Impossibility Theorem [28], which highlights the limitations of a single authority representing a society's preferences. Our governance approach, thus, is designed to foster adaptability, collaborative strength, and strategic depth, positioning survival and success as collective achievements in a constantly evolving game environment.

Algorithm 19: Decide Governance Function

```
Function DecideGovernance():
    Initialise fellowBikers, numberOfFellowBikers and numberOfBadAgents;
    foreach fellowBiker in fellowBikers do
        count numberOfBadAgents based on reputation and threshold;
    if meet Leadership strategy conditions then
        return Leadership;
    return Democracy;
```

12.3.5 Accept/Vote Off

Accept

Based on Condorcet Jury Theorem [15], the commitment to collective cooperation is a concept aimed at maximising interests, and we consistently strive to achieve full capacity within our agent's Mega-Bike (as shown in Algorithm 20). The judgement criteria are based on the principles of the Social Exchange Theory [29] and Social Identity Theory [42]:

- **Social Exchange Theory**

  Social Exchange Theory posits that individuals seek to maximise their rewards and minimise their costs in relationships. To achieve collective objectives, organisational needs are severed as the prime consideration. We aim to cultivate a team with collective objectives, thereby reducing the costs associated with potential divergences and facilitating more efficient goal attainment. Our primary criterion for initial scrutiny involves assessing whether pending agents share the same target color. Additionally, we anticipate that agents with lower energy levels may contribute less energy, consequently emphasising our preference for the inclusion of agents with higher energy levels.

- **Social Identity Theory**

  Social Identity Theory explores how individuals categorise themselves and others into social groups. Reputation serves as a valuable metric reflecting the perception of an agent's role within the team from the perspective of their peers. Concurrently, we take into account the historical experiences of agents, including whether they have a track record as a reputable member or leader. This particular indicator will assume a pivotal role, especially in the context of leadership democracy, being seen as a crucial criterion for selecting leaders. Furthermore, to foster the development of social network connections and enhance individual reputations, our agents incline to trust in embracing new members, particularly addressing the 'first encounters' problems.

Algorithm 20: DecideJoining Algorithm for Agent Recruitment

```
Function DecideJoining(pendingAgents):
    Initialise DecisionMap, ScoreMap and Vacancies variables;
    for each uuid, agent in agentMap do
        score ← score formula based on energy level, reputation and color similarity of agent;
        scoreMap[uuid] ← score;
    rankedScoreMap ← rankByPreference(scoreMap);
    for i from 0 to vacancies-1 do
        decision[rankedScoreMap[i]] ← scoreMap[rankedScoreMap[i]] ≥ threshold;
    return decision;
```

Vote Off

Similar to the 'Accept' decision (Section12.3.5), the decision to kick out an agent is largely contingent upon the agent's reputation, energy level, and the target color (as shown in Algorithm 21). In accordance with Nash equilibrium theory, our assessment indicates that, when an agent's energy level is low, there is a greater inclination towards free-riding behavior. We conduct reputation audits for agents with energy levels below a specified threshold. Once an agent's energy level falls below our defined standard, we are inclined to initiate expulsion. However, we exercise a degree of flexibility in expulsion criteria for agents with a specific target color.

Algorithm 21: VoteForKickout - Decide which agent to kick out based on reputation

```
Function VoteForKickout():
    Get all fellowBikers on bike and initialise the voteResults;
    for agent in fellowBikers do
        voteResults[agentID] ← 0 or 1, based on the reputation and threshold based on strategy;
    return voteResults;
```

12.3.6 Stay or Leave

Stay or Leave

The initial assessment of whether an agent exhibits initiative to change their bike is determined by examining whether the agent has failed to earn any score in the previous iterations or if satisfaction has continuously decreased over the assessed period (as shown in Algorithm 22). This situation can be explained through the lens of Ostrom Institution Theory [48], suggesting a lack of organisational discipline within the Mega-Bike, preventing agents from meeting individual expectations and, consequently, impeding social welfare. Hence, the agent expresses a strong desire to switch bikes.

Algorithm 22: DecideAction for Agent to Change Bike

```
Function DecideAction():
    Initialise selfBikeScore and loopNum;
    for loop i in previous k loops do
        for bikeid, score ∈ loopScoreMap[i] do
            update the selfBikeScore and increase the loopNum count by checking BikeId;
    selfBikeScore ← selfBikeScore / loopNum;
    if selfBikeScore < Threshold and enough energy to jump and selfBike is not the best then
        return ChangeBike;
    return Pedal;
```

Change to Which Bike

Once the decision to change bikes is made, the agent proceeds to rank all Mega-Bikes present. This ranking considers parameters such as the average value of the bikes, the quantity of bikes with the same target color, and the reputation of each bike. It is noteworthy that, guided by a mechanism of self-reflection, the value orientation of agents is subject to situation transformation. The weights assigned to judgment criteria used for ranking will be adjusted. This adaptive process ensures that agents continually optimise their strategies to identify the most suitable organisational alignment over time. (as shown in Algorithm 23)

Algorithm 23: Change Bike Algorithm for Agent to Change Bike

```
Function ChangeBike():
    Get all Mega-Bikes and initialise the bordaScores map and acceptBool map;
    foreach bikeID, Mega-Bike in Mega-Bikes do
        bordaScores[bikeID] ← Score based on Energylevel, Colors, Reputations;
        foreach agent in fellowBikers on our bike do
            reputationSum ← reputationSum + reputation of us for other agents;
        acceptBool[bikeID] ← true or false based on the reputationsSum from other bike;
    Initialise the highestBordaScore and winningBikeID variables;
    foreach bikeID, score in bordaScores do
        if score > highestBordaScore and acceptBool[bikeID] then
            update highestBordaScore and winningBikeID;
    return winningBikeID;
```

12.3.7 Lootbox Preference

Propose Direction

Prior to reaching a conclusive decision, it is imperative to ascertain the potential risk of Awdi pursuit of the Mega-Bike associated with our agent. In the event that there is a possibility of becoming a target for Awdi, a strategic choice is made to tactically retreat in the opposite direction to ensure the safety of the agent and the Mega-Bike.

Subsequent to the confirmation of secure conditions, the process to determine the most desirable Lootbox is initiated. A comprehensive survey of all available Lootboxes on the field is conducted, and a rigorous ranking methodology is applied. The primary criteria for ranking encompass compatibility with the agent's target color and an assessment of the agent's energy level. Moreover, the selected Lootbox location must adhere to predefined safety conditions to guarantee the overall well-being of the agent and the associated Mega-Bike. (as shown in Algorithm 24)

Algorithm 24: Algorithm to Propose Direction for Lootbox Targeting

```
Function ProposeDirection():
    Get all lootBoxes and initialise preferences map and safeAngle;
    safe ← false;
    foreach lootBox in lootBoxes do
        Initialise distance, angleBetweenBikeAndBox and angleBetweenBikeAndAwdi;
        if on bike then
            Update angles and distance based on bike's position and orientation;
        Calculate colorPreference and get agent's energyWeighting;
        distanceBoxAwdi← Calculate distance from Awdito lootbox;
        safe ← true or false based on the angleBetweenBikeAndBox and angleBetweenBikeAndAwdi;
        if distanceBoxAwdi> safe distance found by strategy and safe then
            if energyWeighting > EnergyThreshold then
                Calculate preference based on strategy using color and distance, then store it;
            else
                Consider only energy and survival to calculate preference, then store it;
    overallLootboxPreferences ← Rank lootboxes based on softmax(preferences);
    return First lootbox from overallLootboxPreferences;
```

Final Direction Vote

Diverging from the initial round of proposed directions, the current voting process encompasses the impact of agents' values. Agents now exhibit a preference for trusting those with whom they have had prior interactions and received positive evaluations. Consequently, in the ultimate voting phase, a balanced assessment is conducted, taking into account both the desires of agents and their accrued social capital. (as shown in Algorithm 25)

Algorithm 25: Final Direction Vote in Democracy

```
Function FinalDirectionVote(proposals):
    preferenceScores ← initialize map of uuid.UUID to float64;
    LootboxPreferences ← ProposeDirection();
    foreach lootboxid in proposals do
        preferenceScores[lootboxid] ← preference calculated based on LootboxPreferences and proposals;
    return softmax(preferenceScores);
```

12.3.8 Pedal and Steer Actions

As Rational Choice Theory [68] guides the evaluation of the desirability associated with achieving specific goals. Our agent weights the potential gains against the costs and considering the anticipated contributions of fellow agents. By doing so, our agent aligns its contributions with the strategic imperative of maximising collective benefit, ensuring a judicious investment of effort in pursuits that promise optimal returns.

Parallel to this, the agent's energy level assumes a pivotal role in shaping its contribution strategy. In consonance with strategies of energy-based decision-making demonstrated in Section12.2.4, our agent interprets its energy level as a potent resource. Our agents are willing to make more contributions to collective actions. Conversely, during periods of lower energy reserves, the agent exercises prudence in allocating its efforts, ensuring a sustainable and adaptive approach to task engagement. (as shown in Algorithm 26)

Algorithm 26: Decide Force Algorithm

```
Function DecideForce(direction):
    Initialize forces;
    if GetEnergyLevel() > EnergyThreshold then
        Calculate pedal force based on energy, bike velocity and our preference of direction;
    else
        Pedal with lower force to survive;
    foreach lootbox in all lootboxs in environment do
        Find target lootbox based on direction;
    Calculate distance from bike to Awdiand determine steering angle based on distance and orientation;
    if not in danger then
        Calculate angle towards target;
    else
        Calculate angle away from danger to keep safe;
    Normalize steering angle, set forces and update agent's parameters for self-reflection;
```

12.3.9 Resource Allocation

Our strategy (as shown in Algorithm 27) for resource allocation is grounded in Rescher's Theory of Distributive Justice [66] which takes seven canons into account. We prioritise our own needs, considering both objectives and satisfaction. In situations where our needs remain unfulfilled or our energy levels are below a certain threshold, a predisposition exists to allocate a greater share of resources to ourselves.

The remaining resources are then distributed among other agents who have made positive contributions toward desirable direction, reputation and needs. This approach seeks to balance our own requirements with the recognition and reward of agents demonstrating positive efforts aligned with the stated goals.

Algorithm 27: Resource Allocation Strategy

Function `DecideAllocation()`**:**

Gets all fellowBikers on the bike and initialise the allocationMap;

foreach *agent in fellowBikers* **do**

if *agent.ID == GetOwnID()* **then**

allocationMap[agent.ID] ← Value calculated based on energy level, distribution, score rank;

else

allocationMap[agent.ID] ← Value calculated based on energyLevel, action score, reputation;

return *softmax(allocationMap)*;

The detailed descriptions of the algorithms can be found in the Appendix 12.8.

12.4 Experiment

In the group experiment, agents from all groups were included in the environment to mimic real-world conditions and enable a thorough comparison between the Basebiker model and our group's agent. For a more efficient and accurate experiment, specific environmental parameters were strategically adjusted in our final setup, aimed at speeding up the collection of results while maintaining the integrity of the experimental environment.

12.4.1 Experiment of Strategy Parameters

The experiment is conducted to determine the optimal strategy parameters for our agent. The objective was to ensure that the agent's performance either matches or surpasses that of Basebiker. We plotted agent's energy level and accumulated points that follow the change in threshold parameters on x-axis, being displayed on the y-axis. The outcomes of these tests are presented, with Figure12.3 illustrating the threshold choices in `DecideGovernance`, `DecideJoining` and `VoteKickOut`, Figure12.4 for `DecideAction`, Figure 12.5 and 12.6for determining safe angle and safe distance from the Awdi in `ProposeDirection`. 12.7 demonstrates the optimal choice of the energy threshold which implied what personalities the agent may have. The finalisation of these tests led to the establishment of a final threshold for our strategy, which is summarised in Table12.1.

Figure 12.3: Experiment for Stage 1-3 Threshold

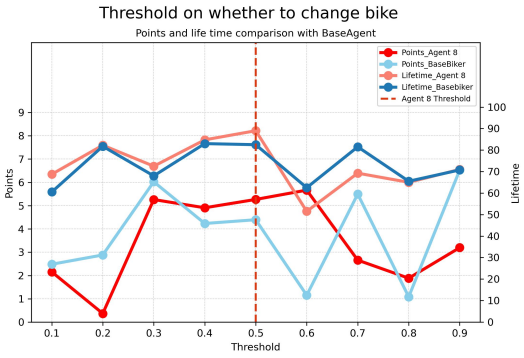


Figure 12.4: Whether to Change Bike

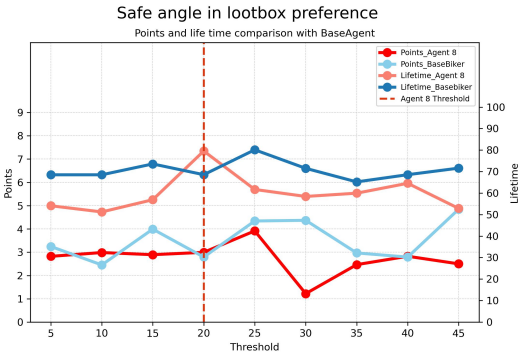


Figure 12.5: Safe Angle in Lootbox Preference

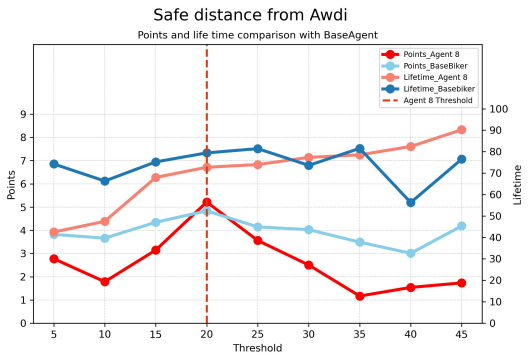


Figure 12.6: Safe Distance from Awdi

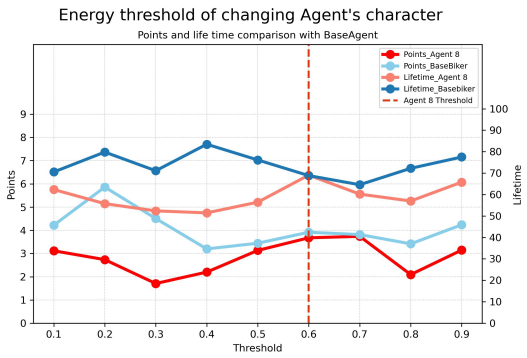


Figure 12.7: Energy Threshold for Behaviour

Parameter	Value
Reputation for Governance Strategy	0.0
Score for Agent Recruitment	0.2
Reputation for Kickout	-0.2
Threshold on whether to Change Bike	0.5
Safe Angle in Lootbox Preference	20
Safe Distance from Awdi	20
Energy Level Threshold for Behaviour	0.6

Table 12.1: Simulation Parameters

12.4.2 Experiment of Agent Strategy

The following diagrams clearly illustrate the superiority of our strategy compared with BaseBiker and other groups' agencies.

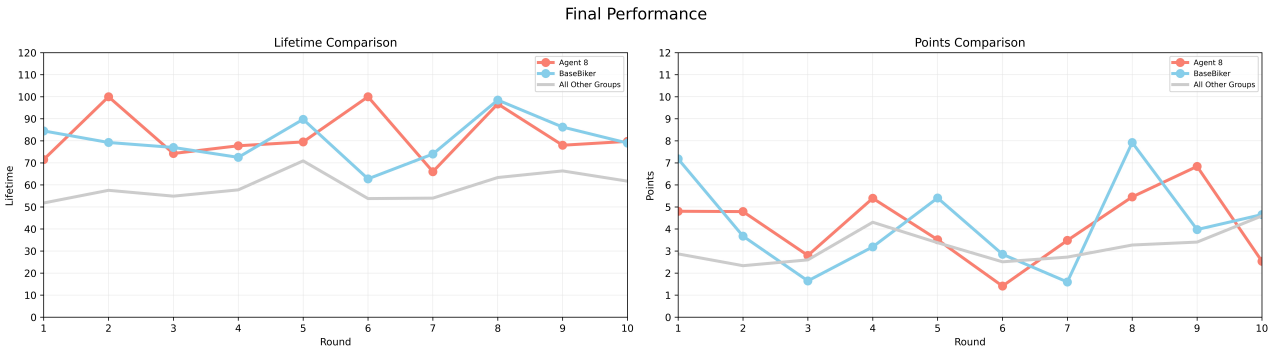


Figure 12.8: Final Performance

The final performance in SOMASWorld is illustrated in the following figure. Given that our final version of the code was released at the last minute, we made significant modifications to the impact of social capital on decision-making. As a result, there are discrepancies between this figure and the results of Group2's tests. Based on the current simulation results, our agents are at an upstream level.

12.5 Future Work

Our agents have adopted strategies that prioritise group interests and democratic principles to varying degrees. However, empirical experiments have revealed that such an approach does not guarantee victory when interacting with other agents. Our future work will involve a critical reassessment of our agent strategies, aiming to refine them from a more discerning perspective. We aspire to strike a balance that not only prioritises group interests but also embraces self-actualisation and maximises social welfare.

A significant challenge in our current game environment is the difficulty in accurately assessing macro performance during agent selection. Metrics such as fairness are challenging to calculate, and evaluating the trustworthiness of communication poses another obstacle. To overcome these limitations, future efforts will

introduce a third-party judgmental perspective to the SOMAS World, creating a more objective evaluation framework and offering insights into nuanced dynamics.

12.6 Project Management

In orchestrating a dynamic and efficient project workflow, our team has embraced the principles of a self-organizing system. This approach has led to the delineation of specific roles tailored to the unique strengths of each member, ensuring a balanced distribution of responsibilities and an unified team dynamic. The structure of our team's organisation is captured in Table 12.2, which details the roles and tasks allocated to each team member. Complementing this structure, the team has committed to a rigorous schedule of weekly meetings to foster continuous communication, track progress, and collaboratively tackle challenges. The frequency and focus of these discussions are documented in Table 12.3, providing a transparent record of our project's evolution and decision-making process.

Responsibility	Charlie Chen	Charlie Li	Nemo Xue	Roy Yan	Nick Zheng	Yuhe Zhang	Kerry Zuo	Alexander Panagiotidis
Team Captain	✓							
MVP Coding	✓	✓		✓				
Strategy Design	✓	✓	✓	✓	✓	✓	✓	✓
Agent Coding	✓	✓	✓	✓	✓	✓	✓	✓
Experiment	✓	✓		✓	✓			
Presentation			✓			✓		
Report	✓		✓			✓	✓	✓

Table 12.2: Team Responsibilities

Time	Theme	Content
02/11	Coursework & Game Loop Understanding	Compile questions for clarification on the project specs
06/11	Team Coordination & Task Allocation	Recap team meeting, assign tasks based on skills and interest
13/11	Lecture Insights & Strategy Formulation	Apply lecture notes to design our project strategy
16/11	Strategy Development & Review	Review team strategy, ensure alignment with project goals
21/11	Agent Coding Phase	Begin coding the agent, consider potential strategy additions
24/11	Code Finalisation & Testing	Complete initial code, start debugging and testing phases
01/12	Game Loop Adaptation, Feature Add-Ons	Adjust code to game loop changes, add new functional features, and find references to support strategy
06/12	Code Review & Finalisation	Review and finalise code, prepare for project submission

Table 12.3: Project Timeline and Activities

12.7 Conclusion

Our group's agent strategy integrates social, psychological, and economic theories, articulating expected capabilities in clear code. By introducing a self-modification agent strategy, we emphasize adaptation and efficiency in dynamic game environments. Rooted in self-awareness and metacognition, our agent autonomously optimizes operational parameters to maximize social welfare while adhering to predefined goals. The game results highlight the strategy's advantages, particularly in a democratic and leadership governance system. Given the diversity of other groups' strategies, we found potential for further enhancing our agent strategy from game interaction.

12.8 Detailed Pseudo Code

Algorithm 28: UpdateReputation Function for Agent8

```
Data: agentsActionsMap, messageReputation
Result: Updated reputations for agents
Function UpdateReputation():
    initialize agentCount as a map of agent IDs to float;
    initialize agentScore as a map of agent IDs to float;
    for i = 1 to number of iterations we recorded do
        foreach agentId, Score in agentsActionsMap[i] do
            if Score is not 0.0 then
                agentScore[agentId] += Score * forgiveness ratio^i;
                agentCount[agentId]++;
            end
        end
    end
    foreach agentId, scoreSum in agentScore do
        if agentId is the ID of the current agent then
            SetReputation(agentId, 1);
        else
            SetReputation(agentId, min(1, scoreSum / agentCount[agentId] +
              messageReputation[agentId]));
        end
    end
```

Algorithm 29: Function to update agent action map

```
Data: Agent8 object bb
Result: Updates bb's agent action map
Function updateAgentActionMap():
    currentLoopAgentActionMap ← new map;
    for agent ∈ all agents on current bike do
        currentLoopAgentActionMap[agent.GetID()] ← agent's energyLevel;
    end
    for i ← 1 to number of loops to record - 1 do
        bb.agentsActionsMap[i] ← bb.agentsActionsMap[i+1];
    end
    bb.agentsActionsMap[10] ← currentLoopAgentActionMap;
```

Algorithm 30: Function to update loop score map

Data: Agent8 object bb
Result: Updates bb's loop score map
Function `updateLoopScoreMap()`:

```
previousDistanceBikeBox ← calculate the distance between bike and target for previous iteration;
currentDistanceBikeBox ← 0.0;
currentDistanceBikeBox ← calculate the distance between bike and target for current iteration;
loopScore ← 0.0;
if we don't get lootbox then
    loopScore ← Score calculated using strategy based on distance change ration, energy loss;
else
    if we get lootbox with our coler then
        loopScore ← higher rate * score calculated based on energy gain;
    else
        loopScore ← lower rate * score calculated based on energy gain;
    end
end
for i ← 1 to number of loops to record - 1 do
    bb.loopScoreMap[i] ← bb.loopScoreMap[i+1];
end
bb.loopScoreMap[10] ← new map;
bb.loopScoreMap[10][current bike] ← loopScore;
```

Algorithm 31: Function to update satisfaction

Data: Agent8 object bb
Result: Updates bb's satisfaction
Function `UpdateSatisfaction()`:

```
alpha ← hyper parameter1;
beta ← hyper parameter2;
distancePrev ← calculate the previous distance between bike and target;
distanceCurr ← calculate the current distance between bike and target;
distanceDiff ← distancePrev - distanceCurr;
energyDiff ← value of energy loss or gain this iteration;
if loss energy and get far away from target then
    bb.satisfaction ← value calculated based on alpha, previous satisfaction, satisfaction ratio;
    if target has the same color as what we need then
        bb.satisfaction increases by ratio;
    end
else
    bb.satisfaction ← lower value calculated based on previous satisfaction, beta;
end
```

Algorithm 32: CreateMessage

Result: Create and send a message based on the agent's current state
Function `CreateMessage()`:

```
if we have low energy then
    Send kindly fake message and ask other agent to help us to survive;
end
if we have lower score than others then
    Send message to ask other agents to help us gain more points;
else
    Send 100% real message to gain trust from other agent;
end
```

Algorithm 33: HandleMessage Function

Result: Handle incoming messages and adjust sender's reputation

Function `HandleMessage():`

```
Check the Message Type, the logic below is based on the Type;
if message contains desired information then
    Add extra reputation to the sender;
else
    if message contains selfish or harmful information then
        Reduce the reputation of the sender;
    end
end
```

Algorithm 34: Decide Governance Function

Result: Determine the governance model based on deliberative and leadership democracy principles.

`// Pre-processing task: Assessing the disposition of fellow agents through effective communication via the Message System.`

Function `DecideGovernance():`

```
fellowBikers ← GetFellowBikers()
numberOfFellowBikers ← len(fellowBikers)
numberOfBadAgents ← 0

foreach fellowBiker in fellowBikers do
    // Logic for deciding voting weight
    if agent's reputation < threshold then
        numberOfBadAgents++
    end
end

if meet Leadership strategy conditions then
    return Leadership
else
    return Democracy
end
```

Algorithm 35: DecideJoining Algorithm for Agent Recruitment

```
Result: Decision Map for Joining Agents
Function DecideJoining(pendingAgents):
    initialise decision as empty map of UUID to boolean;
    initialise scoreMap as empty map of UUID to float64;
    threshold ← GlobalParameters.ThresholdForJoiningDecision;
    agentMap ← UuidToAgentMap(pendingAgents);
    vacancies ← BikersOnBike - numberOfFellowBikers;
    // Calculate the score for each agent and store scores into map
    for each uuid, agent in agentMap do
        if agent's color equals current agent's color then
            score ← higher rate * basic score formula based on energy level and reputation of agent;
        else
            score ← lower rate * basic score formula based on energy level and reputation of agent;
        end
        // store score and set up map
        scoreMap[uuid] ← score;
        decision[uuid] ← false;
    end
    if length of scoreMap equals 0 then
        return decision;
    end
    // rank the agents with map and update decision map use threshold
    rankedScoreMap ← rankByPreference(scoreMap);
    for i from 0 to vacancies-1 do
        decision[rankedScoreMap[i]] ← scoreMap[rankedScoreMap[i]] ≥ threshold;
    end
    return decision;
```

Algorithm 36: VoteForKickout - Decide which agent to kick out based on reputation

```
Result: Returns a map of agent IDs and their vote results
Function VoteForKickout():
    Data: fellowBikers: list of fellow agents
    voteResults ← an empty map;
    for agent in fellowBikers do
        agentID ← agent.GetID();
        if QueryReputation(agentID) < Threshold calculate by strategy based on the environment then
            // Vote to kickout
            voteResults[agentID] ← 1
        else
            // Do not kickout
            voteResults[agentID] ← 0
        end
    end
    return voteResults;
```

Algorithm 37: DecideAction for Agent to Change Bike

```
Data: loopScoreMap, GlobalParameters, currentBike, currentEnergyLevel
Result: Decision to stay on the current bike or change to a new bike
Function DecideAction():
    selfBikeId ← currentBike
    selfBikeScore ← 0.0
    loopNum ← 0.0
    // Calculate total reflection score for current bike
    for loop i in previous k loops do
        for bikeid, score ∈ loopScoreMap[i] do
            if bikeid == selfBikeId then
                selfBikeScore ← selfBikeScore + score
                loopNum++
            end
        end
    end
    selfBikeScore ← selfBikeScore / loopNum
    // Check if we need to change bike
    if selfBikeScore < Threshold calculated by strategy and has enough high energy then
        if The Best bike is not our bike then
            return ChangeBike
        else
            return Pedal
        end
    end
    return Pedal
```

Algorithm 38: Change Bike Algorithm

```
Result: Returns the ID of the best bike to join, the function will be called if we try to jump
Function ChangeBike():
    megaBikes ← Get all bikes from game environment;
    bordaScores ← Initialize a map for Borda scores;
    acceptBool ← Initialize a map for acceptance boolean;
    // set our self bike to true
    acceptBool[GetBike()] ← true;
    foreach bikeID, megaBike in megaBikes do
        bordaScore ← strategy calculation based on Energylevel, Colors, Reputation for agents of bike;
        bordaScores[bikeID] ← bordaScore;
        agentsOnBike ← Get all agents on that bike;
        reputationSum ← 0.0;
        foreach agent in agentsOnBike do
            reputationSum ← reputationSum + reputation of us for other agents;
        end
        if other agents on that bike think we are nice and will to accept us then
            acceptBool[bikeID] ← true;
        end
    end
    highestBordaScore ← 0.0;
    winningBikeID ← Initialize;
    foreach bikeID, score in bordaScores do
        if score > highestBordaScore and acceptBool[bikeID] then
            highestBordaScore ← score;
            winningBikeID ← bikeID;
        end
    end
    return winningBikeID;
```

Algorithm 39: Algorithm to Propose Direction for Lootbox Targeting

Result: Decide which lootbox to target in the first round
Function `ProposeDirection()`:
lootBoxes ← Get all lootboxes
preferences ← Initialize preferences map
safeAngle ← Angle to be find by experience
safe ← false
foreach *lootBox in lootBoxes* **do**
distance, angleBetweenBikeAndBox, angleBetweenBikeAndAudi ← Calculate angles and distance
if *on bike* **then**
Update angles and distance based on bike's position and orientation
end
colorPreference ← Calculate color preference
energyWeighting ← Get agent's energy level
distanceBoxAudi ← Calculate distance from Audi to lootbox
if *angles indicate safety* **then**
safe ← true
end
if *distanceBoxAudi > safe_ distance_ found_ by_ strategy and safe* **then**
if *energyWeighting > EnergyThreshold* **then**
Calculate preference based on strategy using color and distance
else
Consider only energy and survival to calculate preference
end
preferences[lootBox] ← preference
end
else
preferences[lootBox] ← 0.0
end
end
softmaxPreferences ← Apply softmax to preferences
rankedLootBoxes ← Rank loot boxes based on preferences
Store preferences in overallLootboxPreferences for other functions
return *First lootbox from rankedLootBoxes*

Algorithm 40: Final Direction Vote in Democracy

Result: Decide the VotingMap for lootboxList in a democratic way
Function `FinalDirectionVote(`*proposals*`)`:
preferenceScores ← initialize map of uuid.UUID to float64;
`// rerank all lootboxes in MVP`
Call ProposeDirection();
foreach *lootboxid in proposals* **do**
preferenceScores[lootboxid] ← preference calculated by strategy, which combines our preference calculate in ProposeDirection() and preference of other agents ;
end
`// apply softmax function`
softmaxScores ← softmax(preferenceScores);
return softmaxScores;

Algorithm 41: Decide Force Algorithm

Data: direction - UUID of target lootbox
Result: Adjusts force and steering based on energy level and game state
Function `DecideForce(`*direction*`):`

```
Initialize forces
if GetBike() != Nil then
    if GetEnergyLevel() > EnergyThreshold then
        Calculate pedal force based on energy, bike velocity and our preference of direction
    else
        Pedal with lower force to survive
    end
else
    Set pedal force to maximum
end
foreach lootbox in all lootboxs in environment do
    Find target lootbox based on direction
end
Initialize distance to audi
if on bike then
    Calculate distance to audi
end
Determine steering angle based on distance and orientation
if not in danger then
    Calculate angle towards target
else
    Calculate angle away from danger to keep safe
end
Normalize steering angle
Set turning angle and forces
Update agent states and record parameters for self-reflection
```

Algorithm 42: Resource Allocation Strategy

Result: Decide resource allocation based on Rescher's Theory of Distributive Justice
Function `DecideAllocation():`

```
fellowBikers ← Get all agents on our bike
allocationMap ← InitializeMap()
// Iterate over the agents and update the allocation map
foreach agent in fellowBikers do
    if agent.ID == GetOwnID() then
        allocationMap[agent.ID] ← Value calculated by strategy based on energy level, distribution
          for current lootbox, score rank
    else
        allocationMap[agent.ID] ← Value calculated by strategy based on energy level, agent's action
          score for current lootbox, reputation
    end
end
return softmax(allocationMap)
```

Chapter 13: Experiments

13.1 Introduction

The motivation of this section is to explore how the manipulation of world parameters manifests diverse and unique behaviours in the various agents. Within this section, there is an exploration of baseline tuning, the methodology used during this process, and experimentation that is unique to each team's strategy.

13.1.1 Results - Excel Sheet

Due to the nature of the report and to not overload the reader with all the data found, the link to the shared Excel sheet in which all data has been recorded has been attached below: `https://imperiallondon-my.sharepoint.com/:x:/g/personal/as2620_ic_ac_uk/EWaSi1XVl1xEjgPkdWvD5SEBcPYjit8IeTMBTBPv_gjGIQ?e=aPc4Kq`

13.2 Methodology

Initially, the experiments team engaged in a discussion to identify potential experiments that could yield insightful results. During this process, it became evident that multiple inter-playing parameters required consideration.

Defining the tuning space was crucial, taking into account the platform's limitations. This presented a dimensionality problem, arising from the continuous nature of certain parameters and the sheer number of parameters. To address this, the discretization of these continuous parameters became necessary, as well as constraints on the allowable system configurations and the parameters to include.

Once the constraints were established, it was possible to determine the set of feasible configurations. Importantly, the platform is needed either to support adjustments in these parameters or to introduce new parameters. This requirement led to an iterative collaboration between the experimentation and infrastructure teams. These teams communicated regularly, discussing requirements and expectations for the platform's upcoming version. For each configuration, baseline performance and behaviour was established using BaseBiker. Following this, each team's agent conducted experiments to observe variations in behavior.

It is important to note that the BaseBiker always attempts to distribute rewards equally, and has no unique strategy that is dependent on the current world state; in a sense, the BaseBiker has no strategy. In cases where a consensus is made via a vote or decision, the BaseBiker implementation will always adhere to this consensus; foul play does not exist within the BaseBiker implementation.

13.3 Tuning Our Default World

Adjusting the world parameters to establish an Economy of Scarcity was essential for creating a fair and meaningful experiment. The objective was to configure the world in a way that, if agents collaborated and operated cohesively, they could survive, but with a restricted availability of resources to introduce competition for survival. This tuning ensured that during experiments, variations in indicators and metrics would be attributed to a lack of social coordination rather than a shortage of resources.

The table below highlights the final values of the tuned parameters and the corresponding observed metrics. Note that the maximum life expectancy that can be achieved is 100 and the maximum energy an agent can have is 1.

13.4 Individual Experiments

13.4.1 Experimenting with Governance: Deliberative Democracy

Hypothesis and Experiment Description

This experiment evaluates agent performance under a deliberative democracy governance strategy. The expectation is that under a deliberative democracy, the variance in energy and points would decrease, average lifetimes would increase, and equality would be sustained with similar metrics across all agents in the simulation.

Parameter Tuned	Value
Map Size	250 x 250
Number of Agents	56
Number of Bikes	7
Number of Lootboxes	2.5
Initial Agent Energy Level	1
Lootbox Energy	2 to 4
Iterations	100
Rounds	10
Awdi Mass	7
Number of Seats	8
Moving Depletion	0.01
Void Depletion	0.05
Collision Threshold	7
BikerMaxForce	0.8

Table 13.1: Table of tuned parameters and their corresponding tuned values.

Metric Computed Across Game	Value
Average Life Expectancy of All Agents	72.73
Average Energy of All Energy	0.7389
Average Points of All Agents	4.033

Table 13.2: Table of metrics of the game after tuning parameters.

Baseline Results

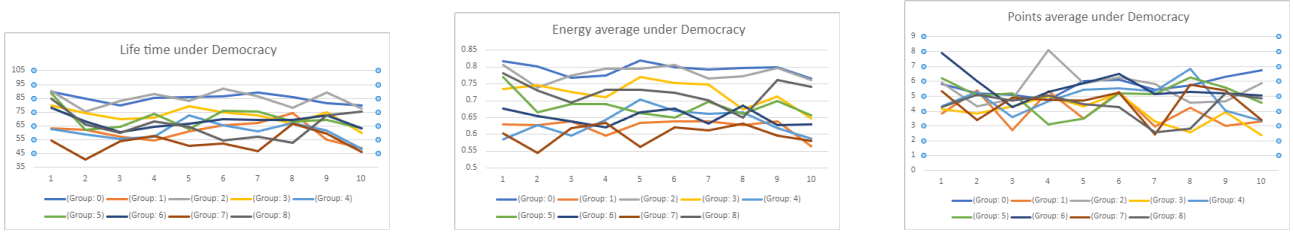
As this is what was calibrated for the default world but rather than the 3x multiplier for the optimal world, the average lifetime is similar to what was found then for the 2.5x multiplier for Lootboxes. The average lifetime is slightly lower than back then due to the nature of the deliberative democracy where there is a cost for voting that affects the energy level of each agent with each vote cast. The average metrics are higher than the ones reported in the parameter tuning from the initial tuning but this can be chalked up to randomness. It is also important to note that the average number of points has also shot up, which indicates we must be hitting more Lootboxes as well.

Additionally, the BaseBiker under this governance randomly decides which fellow agents to kick out during each iteration which means that the agents tend to move between bikes with no real intention. This results in the agents losing more energy rapidly as they keep incurring an energy loss that is attributed to being in the void every other round.

Using the visualiser, we can note there are not very many, if at all deaths from Awdi. Probably due to the predetermined behaviour of the BaseBiker and the Awdi.

This should drastically differ once we run the same experiment with the cohort agents as they have more intelligent behaviours.

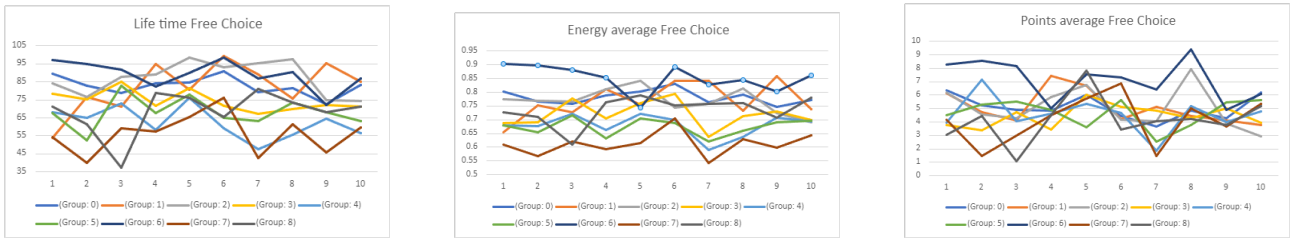
Cohort Results



(a) Life time under Democracy (b) Energy average under Democracy (c) Points average under Democracy

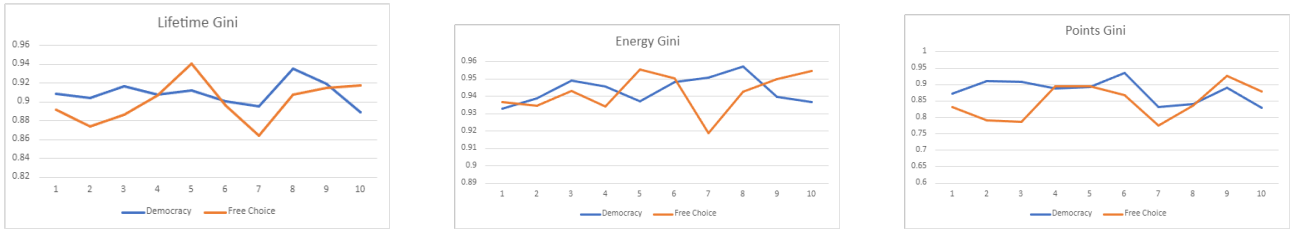
Figure 13.1: Cohort Results Democracy

In the multi-agent scenario, we observe significant changes compared to the BaseBiker situation. Our experiment consists of nine different groups of agents (eight groups of agents plus a BaseBiker, with each agent group



(a) Life time Free Choice (b) Energy average Free Choice (c) Points average Free Choice

Figure 13.2: Cohort Results Free Choice



(a) Lifetime Gini (b) Energy Gini (c) Points Gini

Figure 13.3: Gini index under Democracy

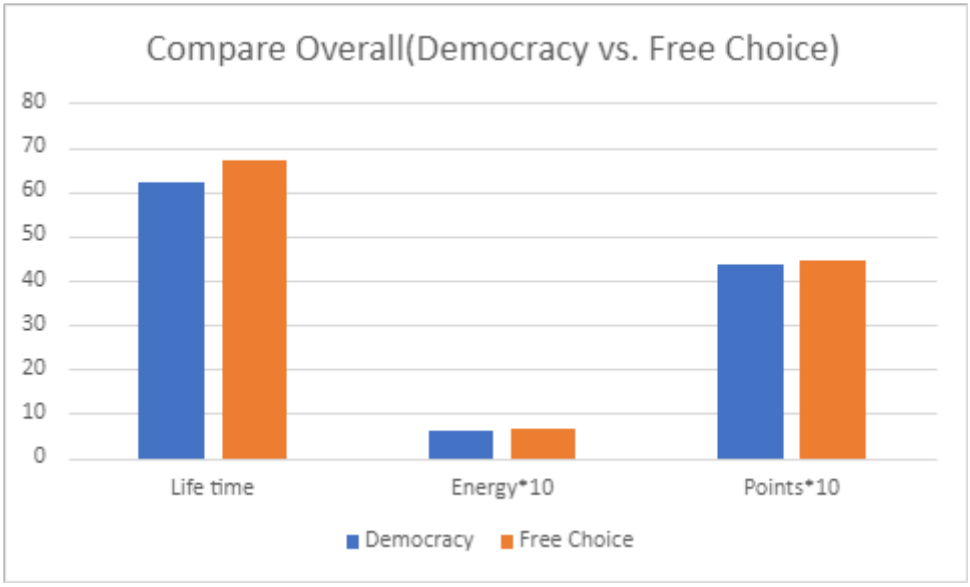


Figure 13.4: Compare Overall(Democracy vs. Free Choice)

comprising six bikers). We evaluate the performance of each agent by recording their average behavior in the same round across five major experiments.

We conduct a control experiment by enforcing a policy of deliberative democracy, and we compare the Gini index in each round. A Gini index closer to 1 indicates a more equal distribution of data. This comparison is used to explore the impact of enforced democracy on the overall societal system. Observing the Gini index across ten rounds shown in the worksheet, it's evident that the enforced democracy has increased the Gini index, leading to a more evenly distributed overall data set. This aligns with our expectations of democratic systems. As illustrated in the "compare overall" graph, compared to normal conditions, all indicators have decreased under forced democracy, a trend consistent with the basic experiment involving only BaseBikers. In the cohort scenario, while some agents show significant differences, the overall mean remains slightly lower than in the Normal case.

Although we are indeed more democratic after enforcing democracy, with more equal distribution across various metrics, the actual gains for everyone (in terms of survival time, energy acquisition, and scoring) have decreased. This is because the BaseBikers cannot determine a clear Lootbox target, leading the groups' Mega-Bikes to advance in oscillating curves rather than a straight line. The unstable acquisition of Lootboxes results in a

shorter overall lifetime and a corresponding decrease in other indicators.

13.4.2 Experimenting with Governance: Leadership Democracy

Hypothesis and Experiment Description

This experiment evaluates the performance of agents under the Leadership Democracy Governance scheme. This means that every agent will have to vote for a leader and there will be one leader elected per bike. This leader will have the power to weigh the votes of other agents however he sees fit.

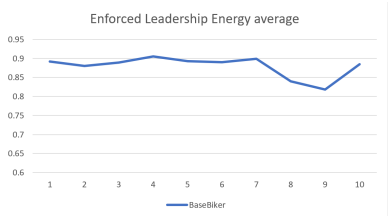
The BaseBiker is programmed to always weigh everyone's vote equally which means that we would expect the results should not be much different than the baseline results for the deliberative democracy baseline results.

Baseline Results

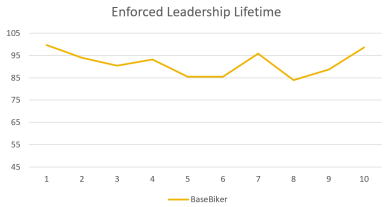
The baseline results are the results obtained when running the experiment using only BaseBiker agents. The results can be seen in the below Table 13.5 and graphs 13.6.

		Metrics		
		Average Energy	Average Points	Average Lifetime
Rounds	1	0.892272	7.259181	99.7
	2	0.880507	7.61951	94.03333
	3	0.88969	6.459961	90.4
	4	0.905371	7.159707	93.2
	5	0.892788	5.256851	85.5
	6	0.890718	5.588412	85.43333
	7	0.899197	6.480847	95.86667
	8	0.840142	4.98643	83.96667
	9	0.81793	6.052596	88.7
	10	0.884984	5.127063	98.6
	Average	0.87936	6.199056	91.54

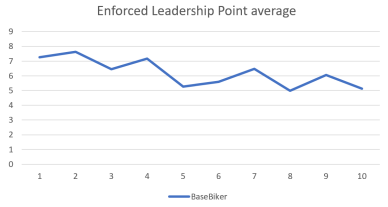
Figure 13.5: Table of BaseBiker baseline results for the leadership democracy experiment



(a) Average Energy per round of BaseBiker Agents



(b) Average Lifetime per round of BaseBiker Agents



(c) Average Points per round of BaseBiker Agents

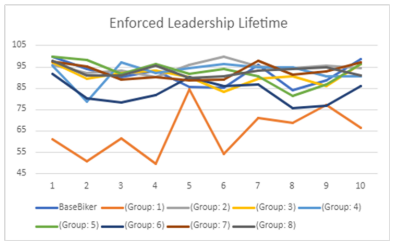
Figure 13.6: Average Metrics per round for BaseBiker agents during the enforced leadership experiment

The results seen in the Graphs 13.6 are very similar to the baseline results for the deliberative democracy experiment which is expected due to the deterministic and simple behaviour of the BaseBiker agents. Comparing the average values across all rounds to the deliberative democracy average results, we realise that there is a very small increase in all three metrics and this could be due many factors such as the energy penalty that is incurred when voting in the deliberative democracy or it could be due to the randomness of the game and BaseBiker since the agents are randomly placed on bikes and the starting distance to Lootboxes is random.

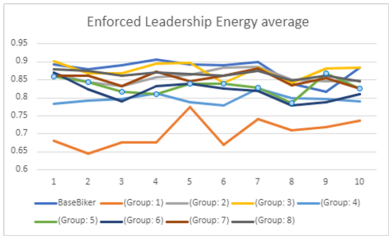
Cohort Results

We conducted another controlled experiment by enforcing leadership as the governance method, and used the Gini index as an indicator for each agent's performance. The results were used to see how all agents behave as dictators in SOMAS World and explore its impact on society.

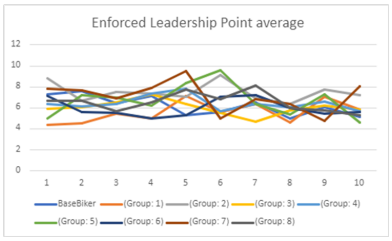
From fig. 13.7a, we can observe that the majority of the agents performed well and had a long lifetime. They have adapted well from democracy to leadership democracy. In fact the majority of the agents have a higher lifetime average in Leadership than in deliberative democracy. From Figure 13.7b, the average energy has



(a) Enforced Leadership Lifetime

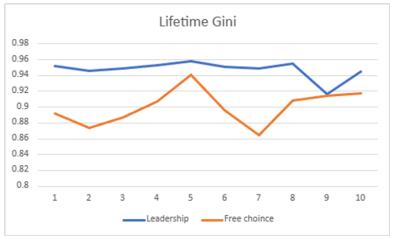


(b) Enforced Leadership Energy average

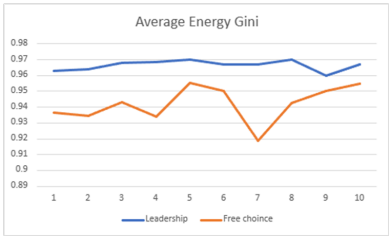


(c) Enforced Leadership Point average

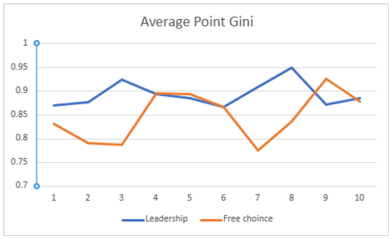
Figure 13.7: Cohort Results Leadership



(a) Lifetime Gini



(b) Average Energy Gini



(c) Average Point Gini

Figure 13.8: Gini Indexs

increased overall for most of the agents in Leadership when compared to deliberative democracy. From Figure 13.7c we can observe that the average points have gone up for almost all the agents when compared to average points for deliberative democracy.

It can be seen from the Figure 13.8, after enforced leadership demoncracy on the agent system, the equality of average lifetime and average energy among all agents is larger than the values obtained for free choice governance. While the gini index of the average point remains approximately the same with the free choice governance. This is probably because the leadership democracy has forced all agents to follow the same decision to some extent which makes the discrimination between agents become lesser. The increment in the gini index of average lifetime and average energy is slightly smaller than dictatorship, and larger than deliberative democracy. The gini index of the average point oscillates in the 10 rounds, while other features are quite fluent. And the average gini index of three features remains approximately the same.

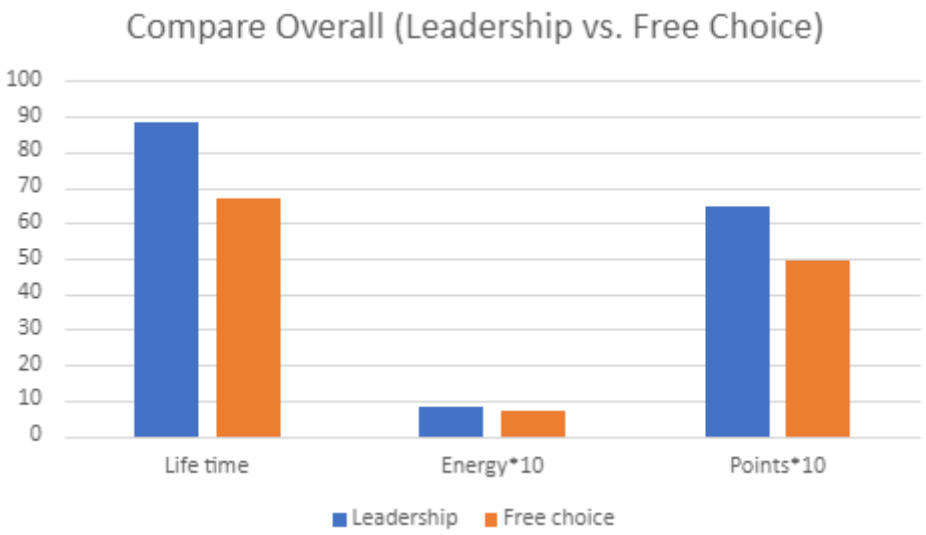


Figure 13.9: Overall Comparison-Leadership vs Free choice

It can be seen from Figure 13.14 that the average lifetime of leadership democracy is significantly larger than free choice governance, while the average energy and points of leadership democracy is slightly larger than free choice governance. The increment of the values is larger than deliberative democracy and lower than dictatorship democracy.

13.4.3 Experimenting with Governance: Dictatorship

Hypothesis and Experiment Description

This experiment evaluates agent performance under a dictatorship governance strategy, predicting harsher conditions would lead to a more polarised distribution of resources and power among agents.

The expectation is that under a dictatorship, the variance in energy and points would increase, average lifetimes would decrease, and equality, possibly measured by the Gini index, would diminish as it is expected that the Dictator is likely to possess most of the resources.

Baseline Results

With dictatorship, life expectancy rose to approx. 90% versus an approx. 70% that was seen in the initial world tuning. Energy remained high, indicating quick, unilateral decisions benefited survival over the short term.

Variations in Lootbox count were then carried out to see how many Lootboxes were now required to achieve a similar life expectancy seen in the original world tuning. Varying the number of Lootboxes showed decreased life expectancy(see Graph 13.11 below), but not due to energy scarcity which would have been the obvious reason as to why.

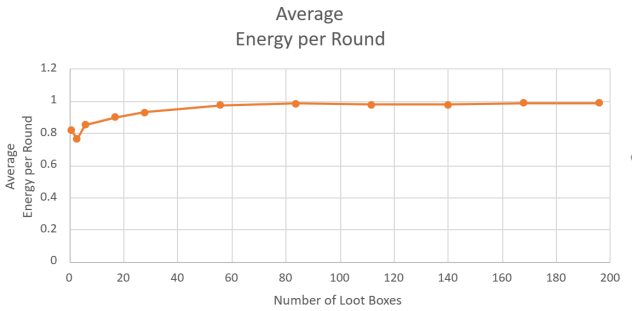


Figure 13.10: Graph showing how the average energy stayed quite consistent despite varying the number of Lootboxes.

Using the Visualiser it was concluded that agents instead became more susceptible to the Awdi, suggesting that while the average energy per agent was plentiful, centralized decision-making made them predictable and vulnerable targets.

This idea can be clearly seen in the image below, where there were only 2 Lootboxes instantiated. Due to the lack of choice, the bikes flocked together, creating a bigger target for the Awdi to hit.

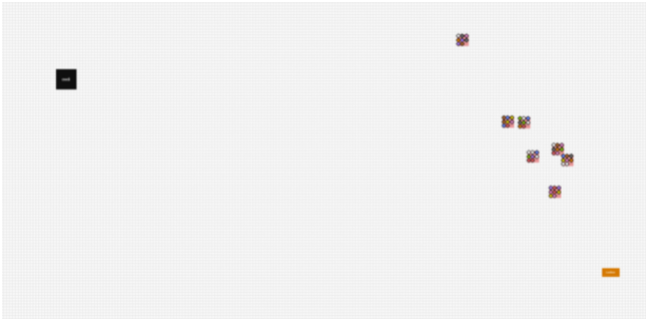


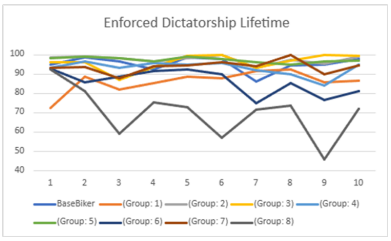
Figure 13.11: Screenshot of Visualiser demonstrating the flocking behavior of the bikes.

This suggests that agents would be able to survive in this Dictatorship even if there was one Lootbox providing there was no Awdi to run them over suggesting a Dictatorship could be an ideal way to socially coordinate. However, it is important to note that the behavior seen in the baseline is likely due to the BaseBiker implementation in which energy and point are distributed evenly and fairly. Incorporating the cohort agents will be more interesting as they will be able to make decisions that exercise the institutional power instilled in being a Dictator. Therefore, the baseline metrics may give a false indicator of how ideal this strategy may be.

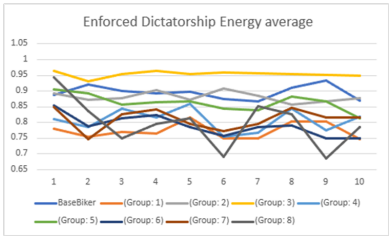
Cohort Results

In the multi-agent scenario, we observe significant changes in the other agents' performance compared to the BaseBiker. Our experiment consists of nine different groups of agents (eight groups of agents plus a BaseBiker, with each agent group comprising of six bikers). We evaluated the performance of each agent by recording their average behaviour in the same round across five major experiments.

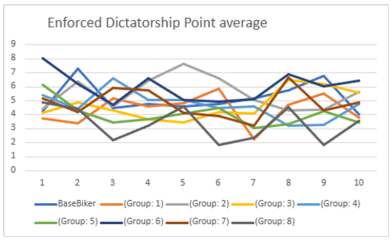
We conducted a controlled experiment by enforcing a policy of dictatorship, and we compared the Gini index in each round. A Gini index closer to 1 indicates a more equal distribution of data. This comparison is used to explore the impact of enforced democracy on the overall societal system.



(a) Enforced Dictatorship Lifetime

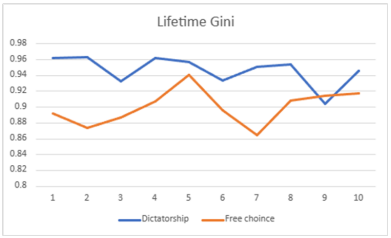


(b) Enforced Dictatorship Energy average

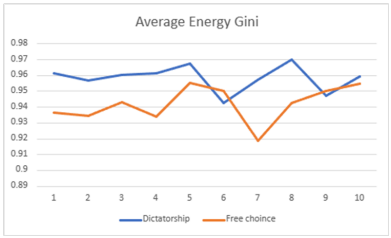


(c) Enforced Dictatorship Point average

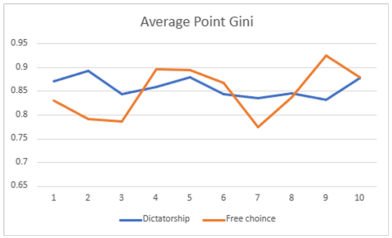
Figure 13.12: Cohort Results Dictatorship



(a) Lifetime Gini



(b) Average Energy Gini



(c) Average Point Gini

Figure 13.13: Gini Indexs

It can be observed from the lifetime graph in Figure13.12a that the lifetime for most agents has increased in dictatorship governance when compared to both leadership and deliberative democracy governance . From Figure13.12b it can be observed that most of the agents also have higher energy levels in dictatorship than in both leadership and deliberative democracy governance. From Figure13.12c, it can be observed that most of the agents in dictatorship have points similar to those obtained in leadership governance. It can be seen from the Figure 13.13, after enforced dictatorship on the agent system, the equality of average lifetime and average energy among all agents is larger than the values obtained for free choice governance. While the gini index of the average point remains approximately the same with the free choice governance. This is probably because the dictatorship has forced all agents to follow the same decision which makes the discrimination between agents become lesser. The increment in the gini index of average lifetime and average energy is slightly larger than leadership democracy, and significantly larger than deliberative democracy. The gini index of three features start oscillating in the 10 rounds, and the average gini index of lifetime decreases gradually, whereas the others remain approximately the same.

It can be seen from fig. 13.14 that the average lifetime of leadership democracy is significantly larger than free choice governance, while the average energy and points of leadership democracy is slightly larger than free choice governance. The increment of the values is larger than deliberative democracy and leadership democracy.

13.4.4 Experimenting with Governance: Choosing Governance Strategy

Hypothesis and Experiment Description

This experiments lets agents choose which governance they would like to be a part of. At each round they vote for a governance and are placed on a bike where their governance of choice is applied.

This experiment is aimed to replicate the game with agents having as much liberty as they can within the set rules. When creating agent strategies teams have tuned their agents to this specific scenario which is why some might say it is the most telling experiment.

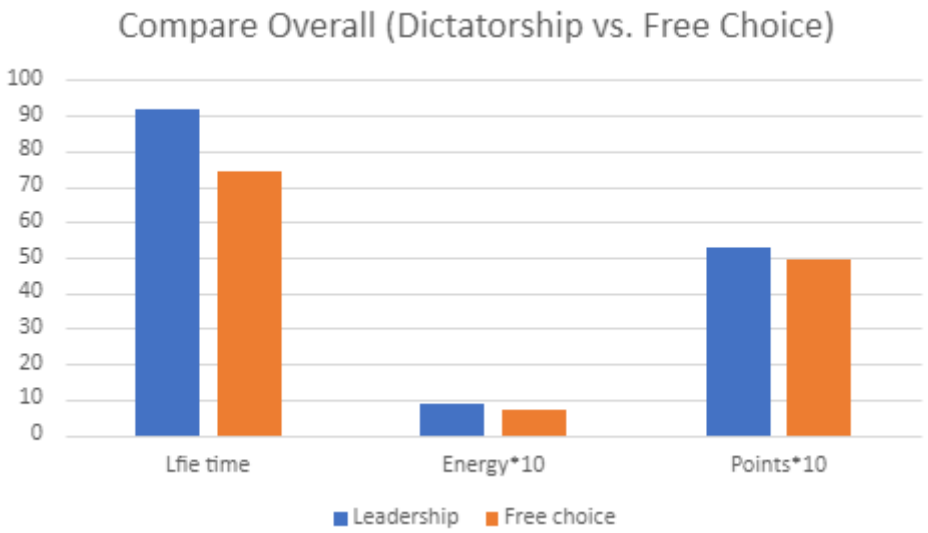


Figure 13.14: Overall Comparison - Dictatorship vs Free choice

It is hard to predict how agents will perform since each team has a different strategy which hadn't been tuned depending on other peoples strategy. This experiment aims to see how well all agents adapt and self-organise with each other and see if agent's collaborate to survive and thrive in SOMAS World.

Baseline Results

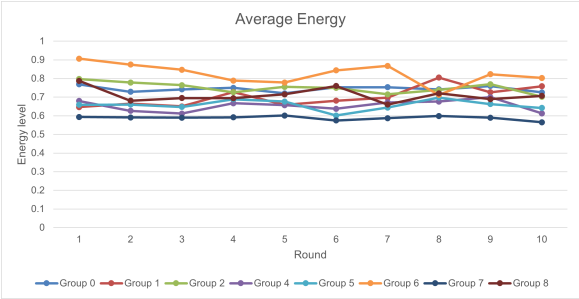
Cohort Results

The cohort results were obtained using 7 agents from each team and BaseBikers (referred to as Group 0 in graphs). However, the agents from team 3 kept crashing and prevented the experiment from completing. As a result, their agents do not figure in our results.The overall average taken for all agents are shown in Table 13.15. The detailed results for each group's agents have been plotted and can be seen in Graphs 13.6.

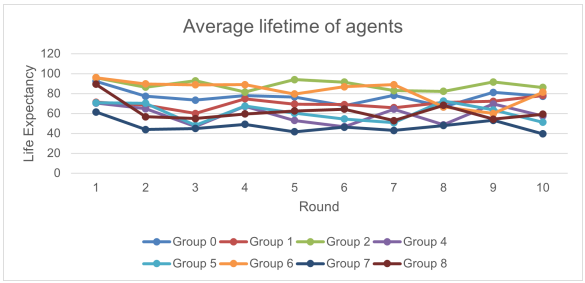
		Trials					
		Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Average
Metrics (per Round)	Average Lifetime	69.3821	64.7787	71.5875	67.4571	68.6982	63.3807
	Variance of Average Lifetime	1.0197	0.9579	0.8768	1.1916	0.7235	0.9539
	Average Energy	0.7201	0.6842	0.7106	0.7064	0.6984	0.7039
	Variance of Average Energy	2.3730	2.3303	2.3639	2.3448	2.3588	2.3542
	Average Points	4.7816	4.3992	4.7605	4.7605	4.7530	4.6910
	Variance of Average Points	9.6277	9.8936	8.3059	12.3626	10.5827	10.1545

Figure 13.15: Table of average metrics combining the data for all group agents

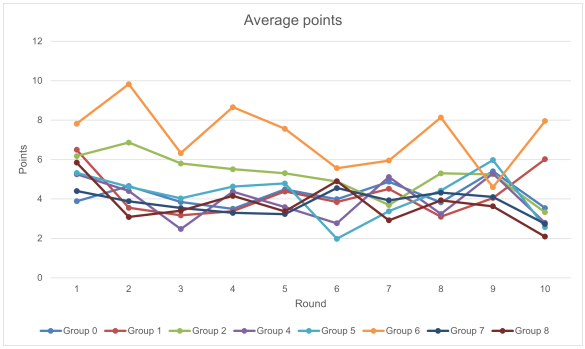
Looking at Table 13.5, we observe that the average lifetime is 63.38 out of 100 iterations per round. This indicates an economy of scarcity, as intended by parameter tuning, where resources are not sufficient for agents to easily survive until the 100th iteration.



(a) Average energy for each group's agents per round



(b) Average lifetime for each group's agents per round



(c) Average points for each group's agents per round

Figure 13.16: Average metrics for each group's agents per round

As a general overview of the Graphs 13.6, all agents perform well and their performance are quite close together. Some set of agents perform a bit better than others which will be commented by the different groups in section 14.

13.4.5 Experimenting with Changing Threats: Energy Scarcity

Hypothesis and Experiment Description

The hypothesis posits that energy scarcity may lead to increased social cohesion or a preference for dictatorship governance due to the elimination of voting costs and the potential for quicker, more decisive action.

The purpose of this experiment was to observe the effects of Lootbox scarcity on the agents' energy conservation and accumulation strategies within the SOMAS environment.

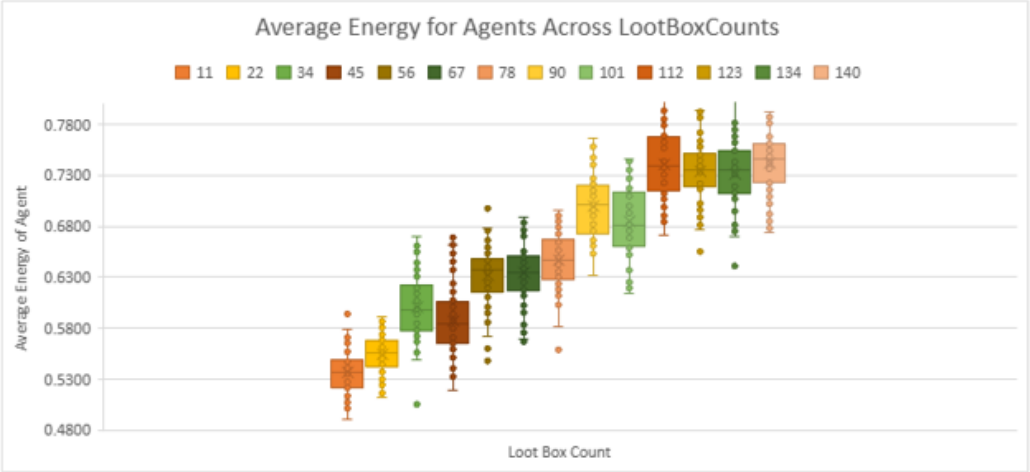
Baseline Results

In optimal conditions with the tuned number of Lootboxes, agents displayed robust average lifetimes and point accrual per round. However, as Lootbox availability declined, a clear decrement in these metrics was observed as can be seen in the graphs below. The median energy levels significantly dipped, particularly noticeable when Lootbox counts were at their lowest. This trend was mirrored in the average points achieved by agents, with peak performance at higher Lootbox counts and a notable decrease as availability diminished.

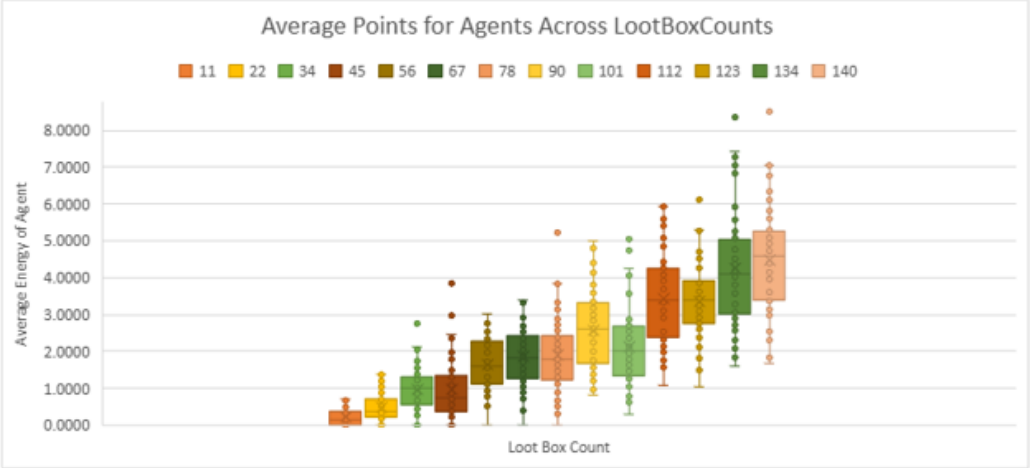
Reduced Lootbox numbers led to a more conservative use of energy among agents, indicated by the reduced variance in energy levels. Despite this adaptation, the widening variance in both lifetime and points per round highlighted an increase in disparities amongst agent performance, reflecting a more cutthroat and uncertain environment as resources become scarce.

Cohort Results

Fluctuations in the quantity of Lootboxes allocated per agent wield considerable influence over both the survival duration and the average points gained by all agents. As the density of Lootboxes increases, there is a significant rise in the likelihood of agents obtaining energy from these boxes, potentially augmenting their chances of securing points. Conversely, a sparser distribution of Lootboxes across the map diminishes the probability of agents accessing both Lootboxes and points. Nevertheless, as the quantity of Lootboxes expands, the average



(a) Average Energy for Agents Across LootboxCounts



(b) Average Points for Agents Across LootboxCounts

variance in energy levels and points among agents proportionally increases.
This scenario mirrors the societal implications of resource distribution within a community. In a society abundant with resources, individuals tend to flourish, with higher opportunity for growth and development. Conversely, limited resources lead to challenges and potentially create disparities among individuals.

13.4.6 Experimenting with Founding Stages: Varying Frequency of Founding Stages

Hypothesis and Experiment Description

The purpose of this experiment was to change the number of founding stages from 0 - n in a given simulation. With an increasing frequency of founding stages, we expect to see an initial round of changing governance type but then see a relative equilibrium being reached where each biker knows more or less which governance best suits them based on their past experiences in previous rounds.

Baseline Results

From Figures 13.19a, 13.19b, 13.19c, 13.19d, 13.19e, 13.19f, we can see that BaseBikers decrease in all their metrics as the rounds increase and the number of iterations in each round decreases except for the average energy of an agent per round. This makes sense because at the start of every round, there is a founding institution and the agents a revitalised with energy so they start anew with max energy (max energy being equal to 1.0). Hence, the shorter the duration of a round, the fewer energy penalties that they have to deal with and the less energy spent on pedalling.

As the BaseBiker is fairly simple in terms of "thinking", unlike the cohort agents, these results make sense as the agents keep switching bikes every other round with every other iteration and pedal towards the nearest Lootbox unless the Awdi is nearby. As a result of this, there aren't any discernable findings in relation to the founding stages affecting the memory/experience retained between rounds as the BaseBiker doesn't remember previous

actions but rather acts on an iteration-by-iteration basis based on its current stats and the environment around it. Additionally, the Awdi did not have the opportunity to intentionally unalive Mega-Bikes and their riders as the number of iterations in a round decreased (aka the frequency of founding stages increased).

This should drastically differ once we run the same experiment with the cohort agents as they have more intelligent behaviours.

Cohort Results

Figures 13.20a, 13.20b, 13.20c demonstrate the statistics we collected for this experiment with agents from each teams.

Analysis of lifespan

Analyzing the line chart depicting the relationship between average lifetime and the frequency of leadership changes, we observe a notable trend. When leadership changed every 10 rounds, there was a marked decrease in the average lifespan of all agents. The reason for that was the low frequency of changing leadership can cause dictatorship. Initially, information is not enough for an agent to learn the strengths and weaknesses of other agents, leadership votes were somewhat random. A selfish or dictatorial leader, coupled with low change frequency, hindered the agents' ability to shift strategies, leading to a significant drop in lifespan. Conversely, when examining higher frequencies of leadership change, ranging from every round to every 10 rounds, we notice fluctuating lifespan values. Initially, lifespans increase as agents can assess each other's reputations based on past performance, enabling more informed leadership choices. However, the subsequent decline results from the inability to timely replace leaders, though the shorter durations prevent drastic reductions in average lifespan. Particularly, changing leadership every round did not yield the highest lifespan for agents. This was primarily due to the instability caused by frequent changes, leading to a lack of cohesive strategy. Groups struggled to quickly adapt to these changes, and the constant shift in objectives resulted in inefficient energy use, ultimately affecting the overall lifespan of the agents.

The point scores for each agent exhibit a pronounced downward trend when the frequency exceeds 10. This trend can primarily be attributed to the shortened lifespans of the agents. With limited time available, agents struggle to collect Lootboxes, resulting in lower point tallies.

Analysis of Points receive

Conversely, at frequencies ranging from 1 to 10, there is an initial increase in points, followed by a decline. This fluctuation might be linked to the early frequent changes in leadership, leading to a lack of cohesive strategy and less efficient energy utilization. Consequently, this inefficiency translates to fewer points. However, as the process evolved, an optimal frequency for leader elections was established. This balance ensured strategic stability while allowing for necessary adjustments, enabling the agents to accumulate more points effectively.

Analysis of Energy remain

The average energy level of agents is exhibiting a noticeable upward trend. This phenomenon primarily occurs because as the frequency increases, the lifespan of each agent is significantly reduced, resulting in agents living for only a few rounds. Consequently, this leads to a higher average energy level

13.4.7 Experimenting with Knowledge Management: Varying ACL

Description

In SOMAS World, much like our reality, agents are not all-knowing. Crucial decision-making information is not always readily accessible to them. However, they acquire this vital information through the communications of talkative agents. These communicative agents generously share information or misinformation regarding their actions with everyone. These actions include pedalling, turning, braking, voting allocations, and target Lootbox.

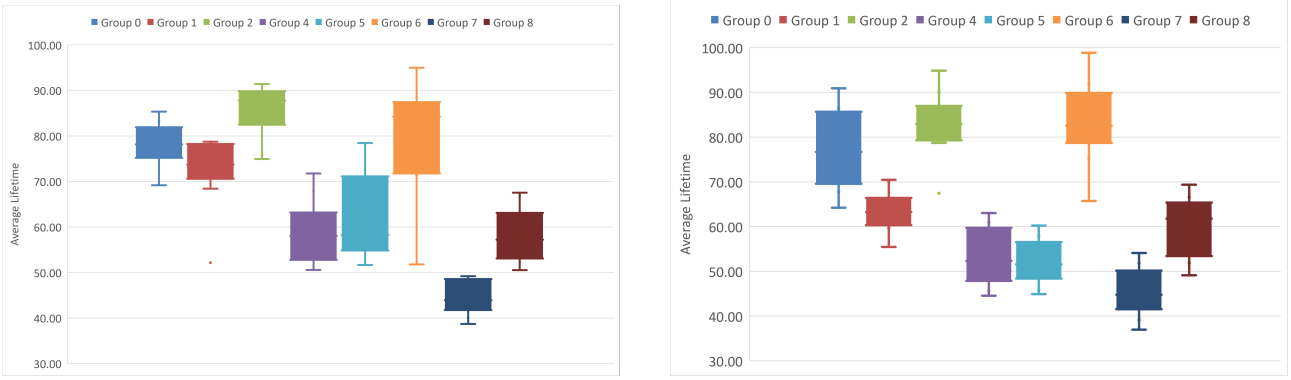
The simulation was run with and without messages being active. It was also run without a baseline given that BaseBiker does not implement messages. The BaseBikers are represented as Group 0.

Hypothesis

Under the assumption that all agents are intelligent and prioritize self-interest. Each agent will employ alternative strategies to gather information, such as observing the energy levels of other agents to infer their pedalling. Consequently, these agents should minimally rely on broadcasted information from others. A critical piece of information for a smart agent would be detecting braking actions. It is posited that a smart agent would neither

engage in braking nor broadcast any intention to brake, as doing so would neither align with self-preservation nor competitive advantage.

Cohort Results



(a) (b)

Figure 13.21: Average lifetime with Messaging (a) No messaging (b)

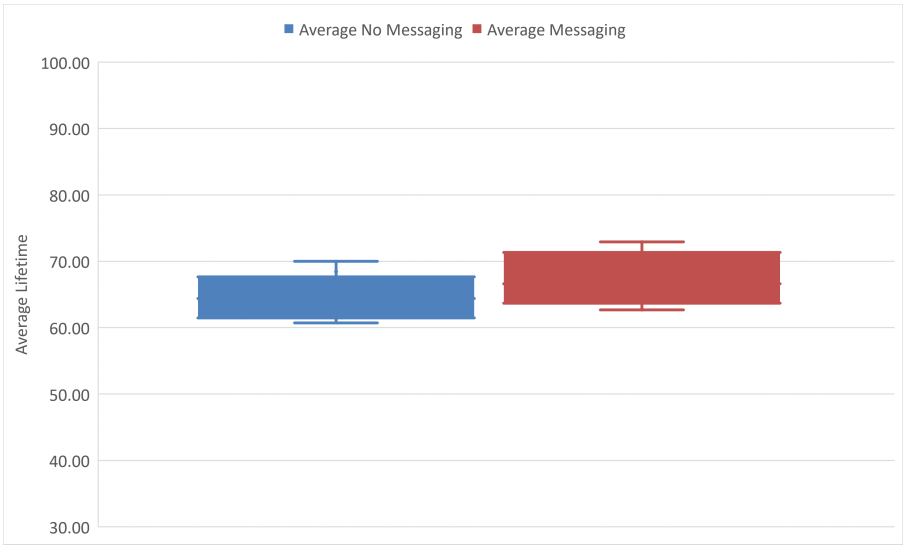
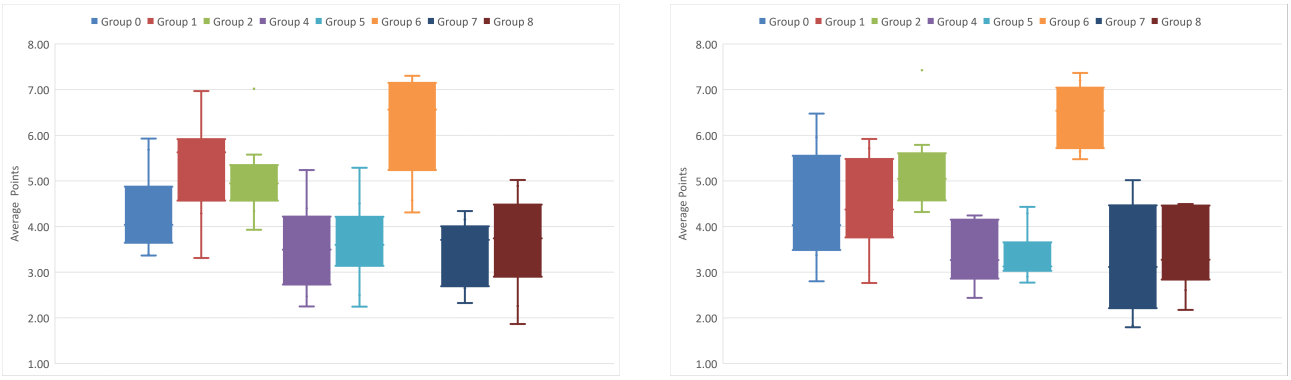


Figure 13.22: Combined Average lifetime messaging vs no messaging



(a) (b)

Figure 13.23: Average Points with Messaging (a) No messaging (b)

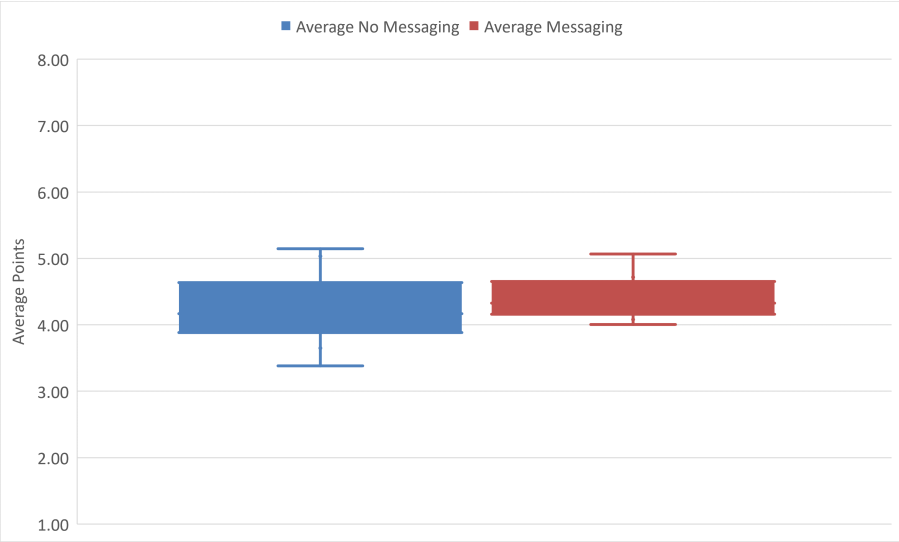
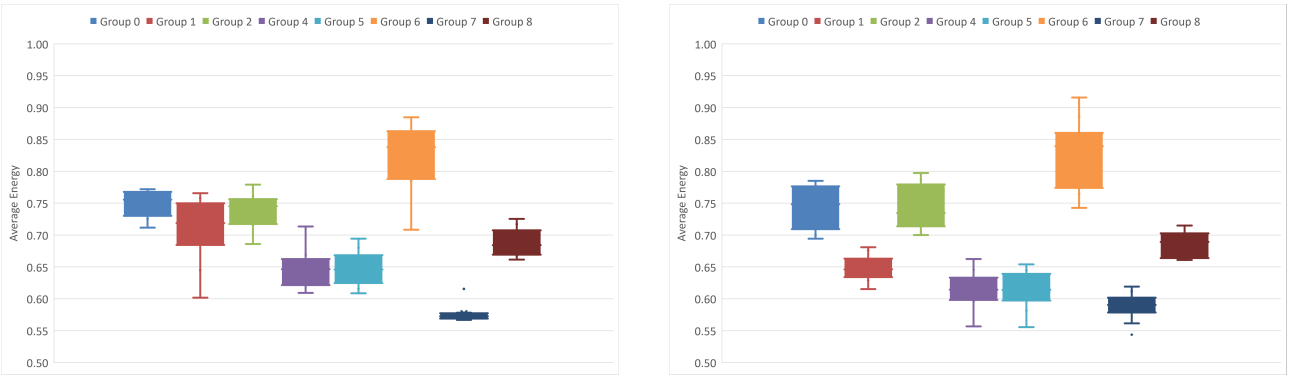


Figure 13.24: Combined Average Points messaging vs no messaging



(a) (b)

Figure 13.25: Average energy with Messaging (a) No messaging (b)

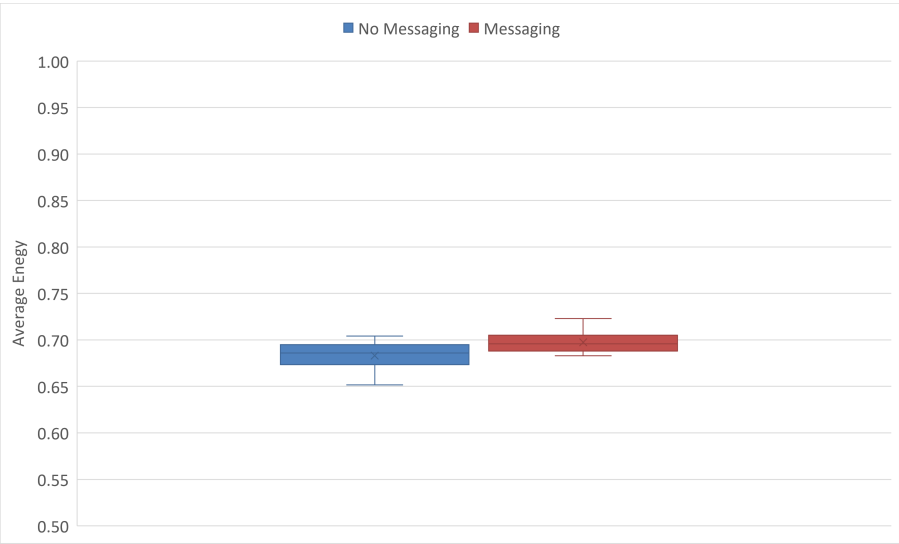


Figure 13.26: Combined energy lifetime messaging vs no messaging

As can be seen in Figures 13.21a to 13.26, agents consistently demonstrate improved performance with the use of messaging, particularly in terms of energy levels. This improvement suggests that several agents are effectively using the transmitted messages to make more informed decisions. Additionally, the nature of the

messages implies that agents are likely sharing truthful information about their actions, possibly as a strategy to build trust with other agents. However, not all agents seem to depend on these messages, or at least not to a significant extent.The overall impact of eliminating messaging is relatively modest, as indicated by the relatively minor impact observed when messaging is removed. Nonetheless, due to the limited number of iterations in this study, drawing a definitive conclusion about the full impact of messaging remains challenging.

13.4.8 Managing Relationships with Difficult Members: Adding Narcissist Agents

Experiment Description

Managing relationships within any group can indeed be challenging, and this becomes particularly complex when dealing with members who exhibit narcissistic traits. Narcissists typically display a heightened sense of self-importance, engage in attention-seeking behaviors, manipulate others for personal gain, and often show a significant lack of empathy towards their peers.

Individuals with these narcissistic tendencies tend to score distinctively in the Five-Factor Model of Personality, also known as the "Big Five" personality traits[1]. These traits include:

- **Low Agreeableness:** Narcissistic individuals often score low in agreeableness, indicating a lack of warmth, trust, and cooperativeness. They may be more competitive rather than collaborative, which can lead to conflicts and challenges in group dynamics.
- **High Conscientiousness:** Surprisingly, some narcissists may exhibit high levels of conscientiousness, characterized by a sense of duty, ambition, and sometimes an overemphasis on status and achievement. This trait can drive their desire for success and recognition.
- **High Neuroticism:** This trait involves emotional instability and a tendency towards anxiety, moodiness, and irritability. In narcissists, this could manifest as sensitivity to criticism and a propensity for defensive or aggressive reactions.
- **High Extraversion:** Narcissists often score high in extraversion, which aligns with their need for attention and admiration. They are typically outgoing, assertive, and often seek the spotlight.
- **High Openness:** This trait involves a high level of creativity, curiosity, and a willingness to explore new ideas. Narcissists may use these characteristics to their advantage in manipulating situations or people.

Adding NPC Narcissists into the World

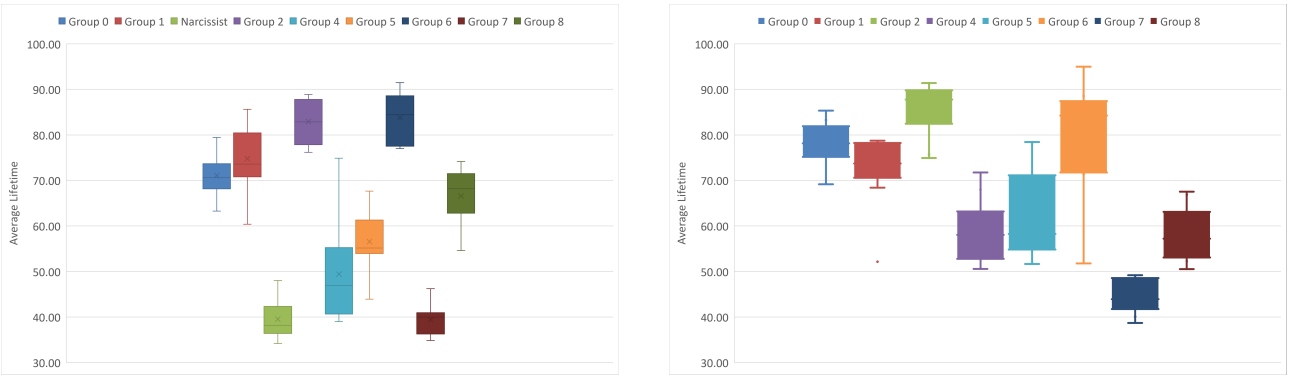
We modify Group 7's agent in a simulated environment to embody the key traits of narcissism as detailed previously as group 7's agent uses the Five-Factor Model of Personality. This modified agent, henceforth referred to as the "Narcissist Agent," is programmed to exhibit high levels of extraversion, conscientiousness, neuroticism, and openness, along with low agreeableness. These traits are calibrated to reflect the typical behavioural patterns of a narcissistic individual in a group setting.

In addition to these personality adjustments, a unique behavioural trait is introduced to the Narcissist Agent: the refusal to 'pedal,' metaphorically representing a reluctance to contribute to the group's collective effort. This behaviour is rooted in the narcissist's grandiose sense of self-importance and a belief that they are above menial tasks. The Narcissist Agent is programmed to act as if it has contributed significantly to the group's efforts, despite its lack of actual participation. This behaviour aims to mimic the narcissist's tendency to seek credit without equivalent effort, further complicating group dynamics and task completion.

Hypothesis

The primary objective of these modifications is to observe and analyze the impact of the Narcissist Agent's behavior on overall group performance. This agent is characterized by a refusal to share workload, high self-regard, and manipulative tendencies. The hypothesis posits that the presence of such an agent will lead to a decline in the group's average lifetime, points, and energy levels. These outcomes will offer critical insights into the group's ability to adapt and self-organize, particularly in strategies to mitigate the negative impact of the Narcissist Agent, potentially by removing such agents from critical roles to avoid detrimental effects on the group's metrics.

Results



(a) (b)

Figure 13.27: Average lifetime with Narcissists (a) without narcissists (b)

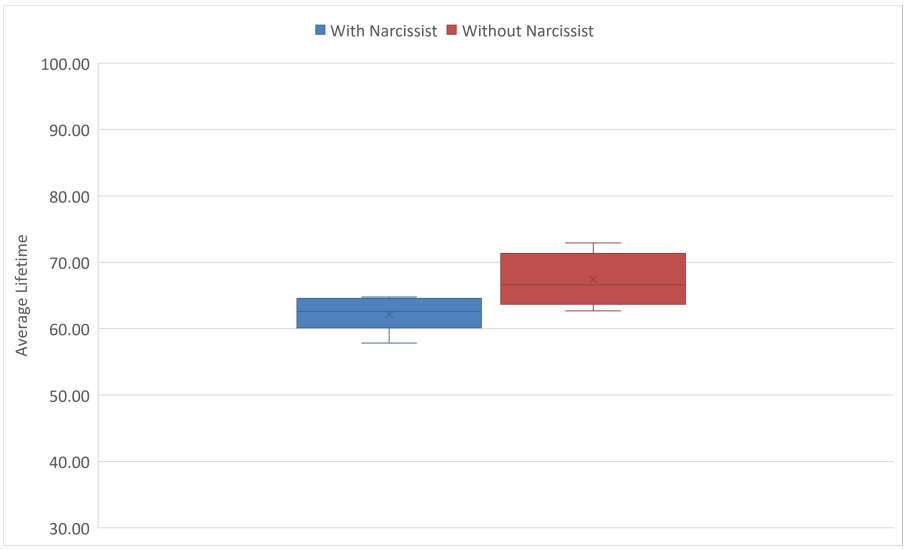
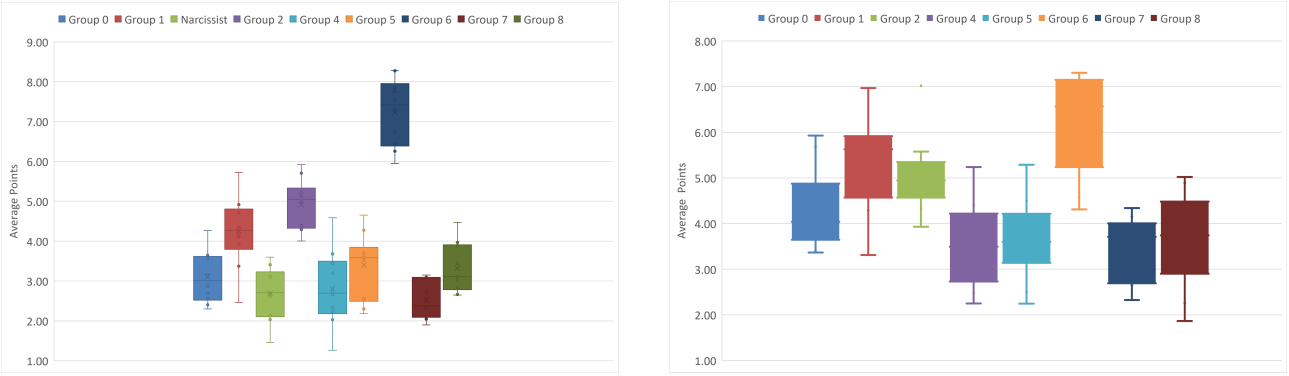


Figure 13.28: Combined Average lifetime with narcissist vs without narcissist



(a) (b)

Figure 13.29: Average Points with Narcissists (a) without narcissists (b)

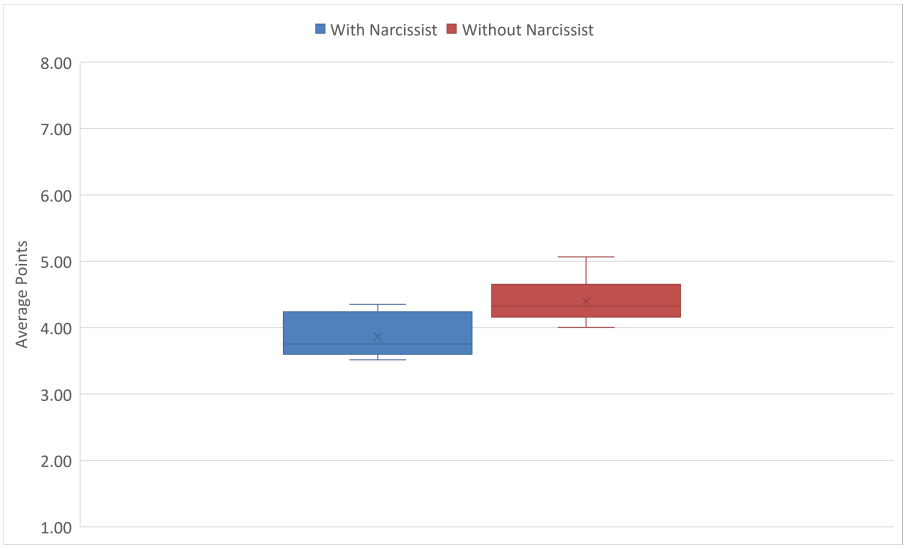
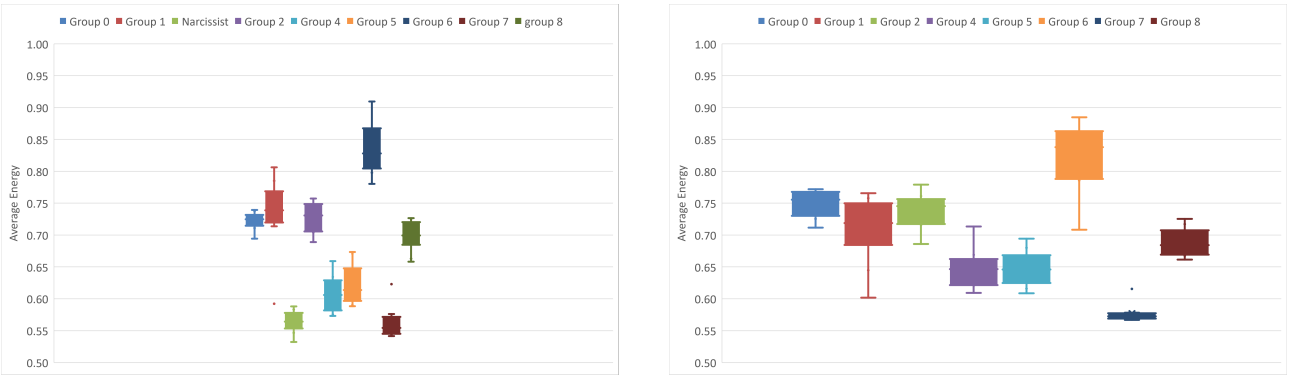


Figure 13.30: Combined Average Points with narcissist vs without narcissist



(a) (b)

Figure 13.31: Average Energy with Narcissists (a) without narcissists (b)

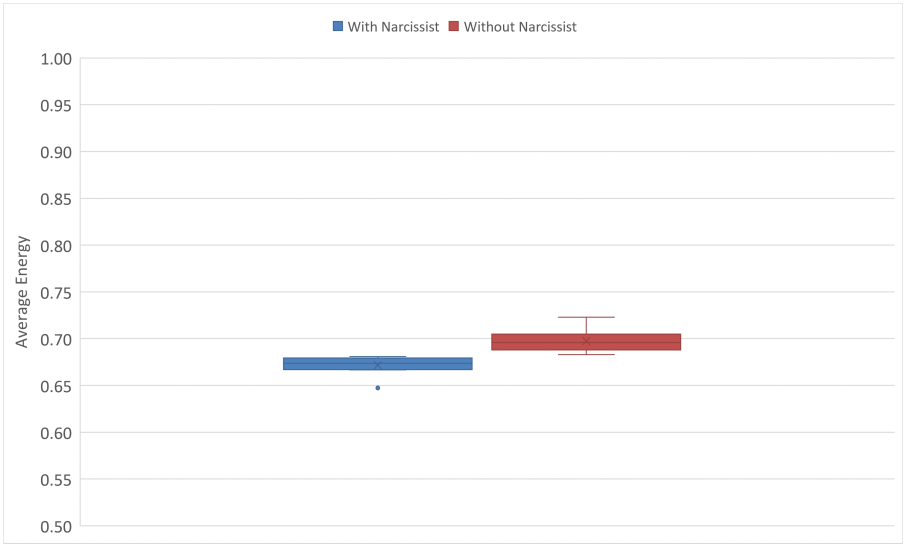


Figure 13.32: Combined Energy with narcissist vs without narcissist

Utilizing Figures 13.31 to 13.32, we observe that the narcissist agent's performance is akin to that of Group 7's agent, which served as its basis. Generally, it ranks low across the three evaluated metrics: average lifetime, average points, and average energy. Consistent with our hypothesis, agents demonstrate improved lifetime met-

rics in the absence of narcissistic agents. However, the impact on points is negligible, whether narcissists are present or not.

Interestingly, agents exhibit a slight improvement in average energy when narcissistic agents are part of the group. This phenomenon could be attributed to two potential reasons: Firstly, agents might recognize the presence of an unproductive 'bad' biker, i.e., the narcissist, who consistently avoids pedalling. This early recognition might lead them to transition sooner to other bikes, where their productivity is enhanced. Secondly, as indicated in Figure 8, the presence of a narcissist might lead to earlier agent 'deaths', resulting in agents retaining more energy at the end of their lifecycle.

In summary, the inclusion of narcissistic agents does not significantly deteriorate the group's overall performance. However, to draw a more definitive conclusion, additional simulations with a greater number of narcissistic agents are necessary.

13.4.9 Investigating the Optimal Dynamics and Resource Equilibrium for Agents

Hypothesis and Experiment Description

The objective is to refine the parameters of the SOMAS environment, based on the results averaged over 10 gameplay iterations. Our primary goal is to enhance the diversity and distinctiveness of the agents' behaviors and outcomes within the game. Specifically, this entails shifting our focus from simply optimizing average lifetime, energy, and point metrics to increasing the standard deviation across these parameters. This adjustment aims to introduce a richer and more varied set of dynamics, thereby fostering a more nuanced and engaging gameplay experience that better accentuates the unique characteristics of each agent.

Experiment Methology

Simulations employing a diverse array of hyperparameters were executed, specifically focusing on linearly distinct instances that accentuate variables such as Moving Depletion, the Number of Lootboxes Per Agent, and Limbo Depletion, which are hypothesized to have significant effects on resource equilibrium. A systematic parameter sweep was conducted, prioritizing parameters according to their influence. During this process, we meticulously gathered statistics to observe the evolving patterns of both the mean and standard deviation for key metrics: Agent's average lifetime, energy, and points. Our objective is to optimize the standard deviation for each parameter to ensure variability, while simultaneously maintaining a mean that is within an acceptable range. Upon the completion of the parameter sweep for one variable, we identify a practical range for that variable. This range is then integrated with the next variable of interest in a sequential sweep, thereby allowing us to refine our model iteratively and enhance its predictive robustness.

Baseline and Tuned Parameters

Parameters	Default Parameter	Tuned Parameter
Moving Depletion	0.01	0.05
Number of Lootboxes Per Agent	2.50	4.50
Limbo Depletion	0.05	0.075
Biker Max Force	0.80	0.80
Awdi Mass	7.00	7.00
Deliberative Democracy Penalty	0.05	0.025
Leadership Democracy Penalty	0.025	0.0125

Baseline and Tuned Results

Metrics	Default Parameter	Starting Point for 0.05	Tuned Parameter	Diff
Lifetime	64.72	40.07	61.98	-2.74
Lifetime std	15.31	8.95	13.21	-2.10
Energy Average	0.67	0.56	0.66	-0.01
Energy std	0.06	0.06	0.07	0.01
Points Average	3.91	1.96	5.71	1.80
Points std	1.61	0.91	2.27	0.66

The origin parameter indeed appears optimal for the world populated by fundamentally selfish bikers, as evidenced by the experiments you conducted. These bikers are characteristically selfish, consistently prioritizing resource allocation for themselves at a maximum level.

Nevertheless, observations from experiments involving agents from all groups indicate a shift in the agents' objective from "finding the best pedalling strategy" to "securing the maximum resources each round." This is because the energy loss per round is almost uniform, with negligible variations arising from the pedalling process. Such a scenario is counterintuitive, as the ideal state would entail a minimal, fixed energy loss from voting, with the primary depletion stemming from other activities. This would motivate agents to refine their strategies in aspects other than resource allocation. By ensuring a variable energy loss tied to diverse behaviors, the desired level of uncertainty within the SOMAS World is maintained.

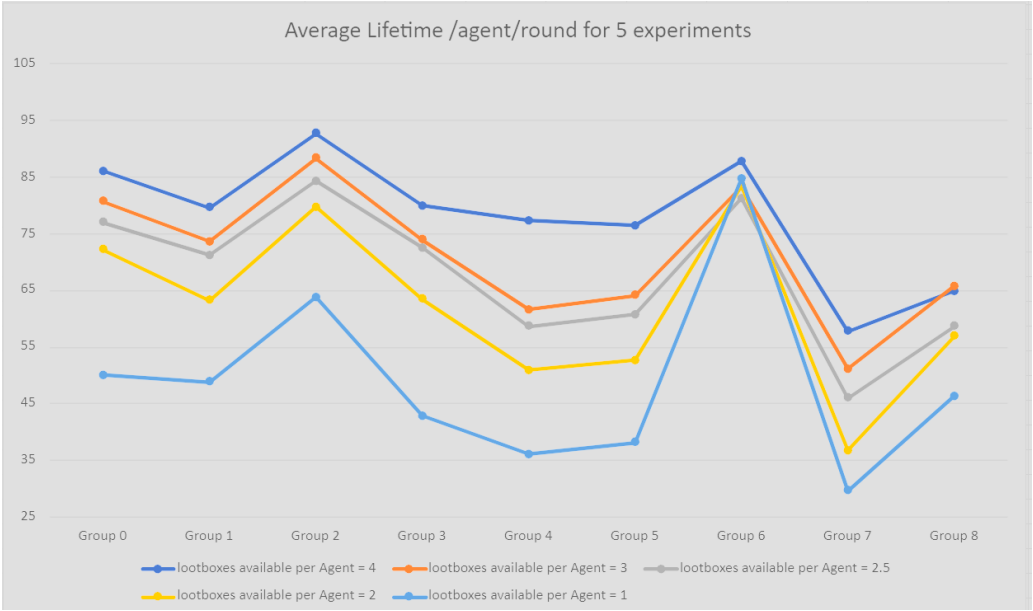
Therefore, the most self-centered agents, who are singularly focused on amassing the most energy during resource allocation—much like the BaseBiker—end up prevailing. This is contrary to our expectation that agents with the most well-rounded strategy should win. After all, the influence of their actions (energy loss) on other processes is trivial. (Intuitively, the bulk of energy loss should result from pedalling rather than voting.)

Conclusion

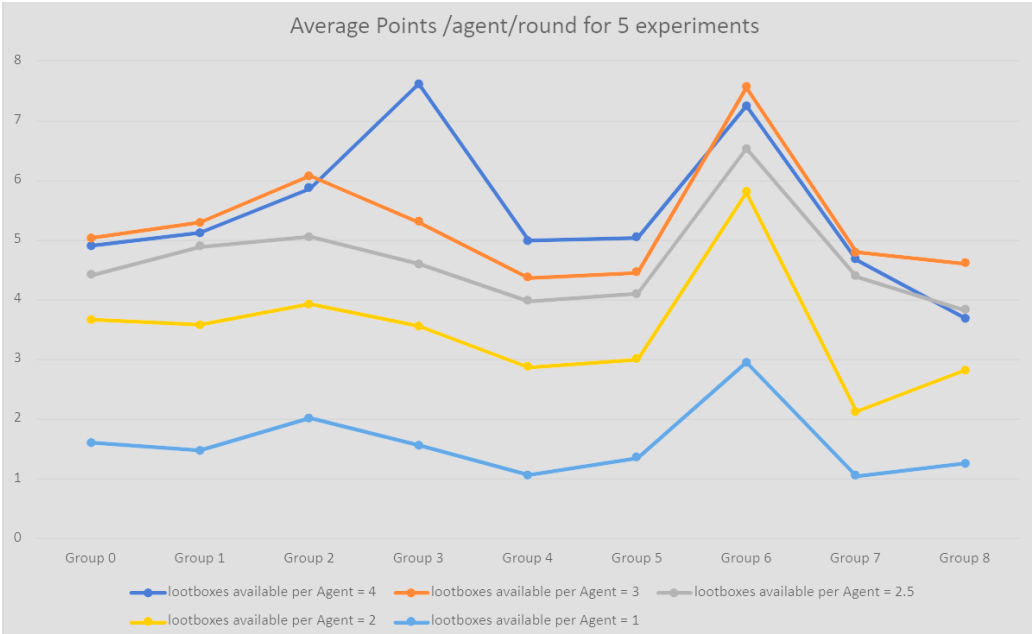
In the conducted experiment within the SOMAS framework, the adjusted parameters revealed a marked trend: agents operating on self-interest, notably exemplified by the BaseBiker, consistently outperformed their cooperative counterparts. This outcome diverges from the intended dynamics of the SOMAS environment, indicating a need for more robust integration of justice-centric parameters and enhanced system transparency.

According to Social Exchange Theory, which posits that individuals conduct cost-benefit analyses in social interactions to maximize personal gains, the minimal repercussions for selfish behaviors in the current setup have skewed the agents' strategies towards self-interest. This is further exacerbated by the limited scope of observable data, primarily restricted to energy consumption, and the presence of misleading information within communication channels, complicating the effective identification and penalization of selfish actions.

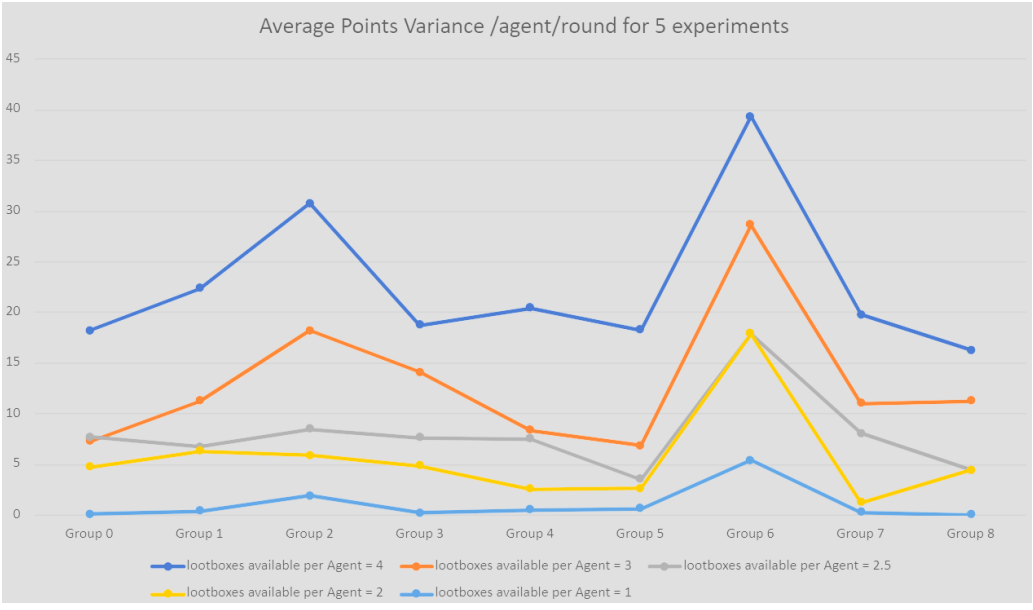
The resulting scenario mirrors the principles of the Tragedy of the Commons and Rational Choice Theory, where individual gain is prioritized over collective welfare, suggesting that in environments with opaque justice mechanisms, a self-serving approach becomes the most rational strategy. The experiment's findings underscore the importance of incorporating Equity Theory, which emphasizes fairness and equity in resource distribution, to counteract the inclination towards selfish strategies and promote a more balanced, cooperative, and sustainable ecosystem within the SOMAS framework.



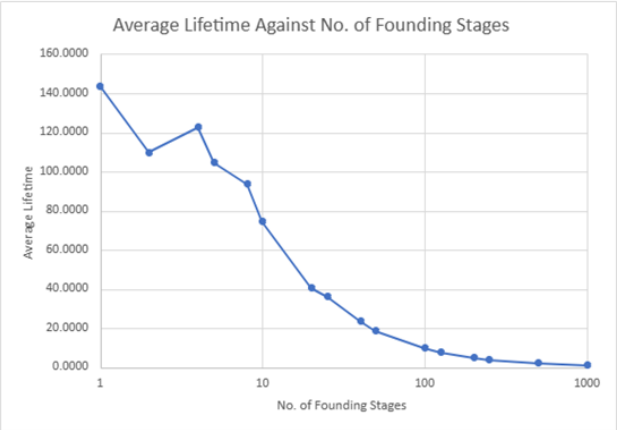
(a) Average Lifetime



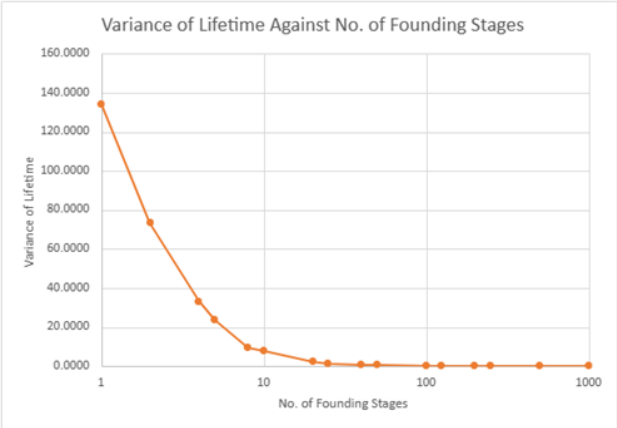
(b) Average Points



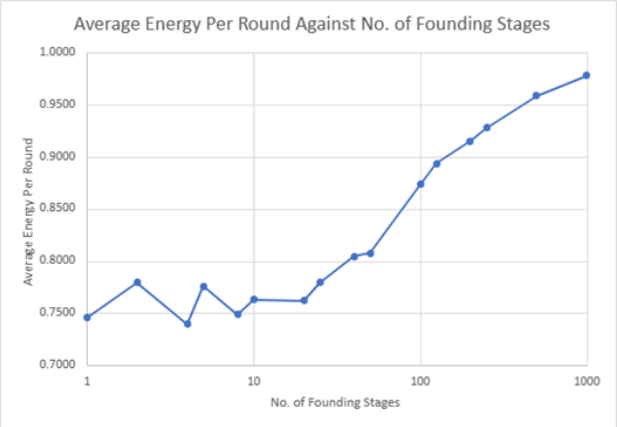
(c) Average Point Variance



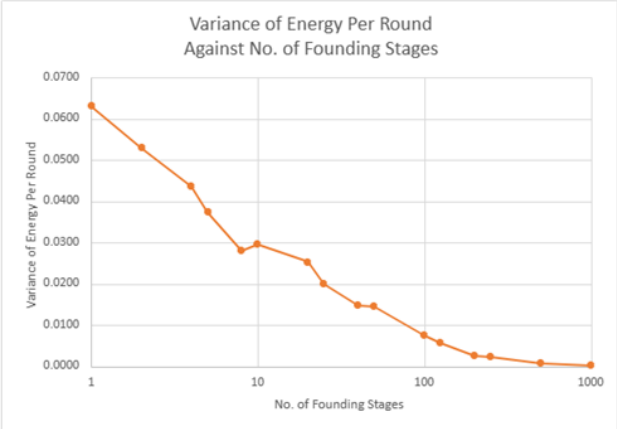
(a) Image 1



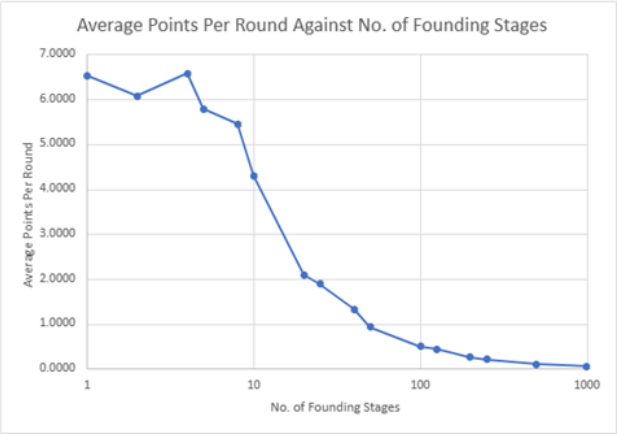
(b) Image 2



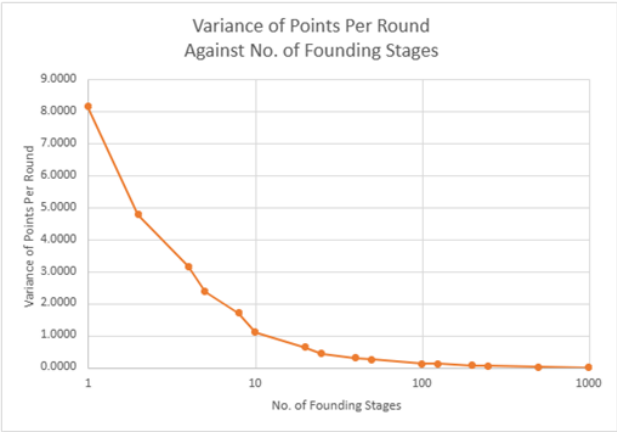
(c) Image 3



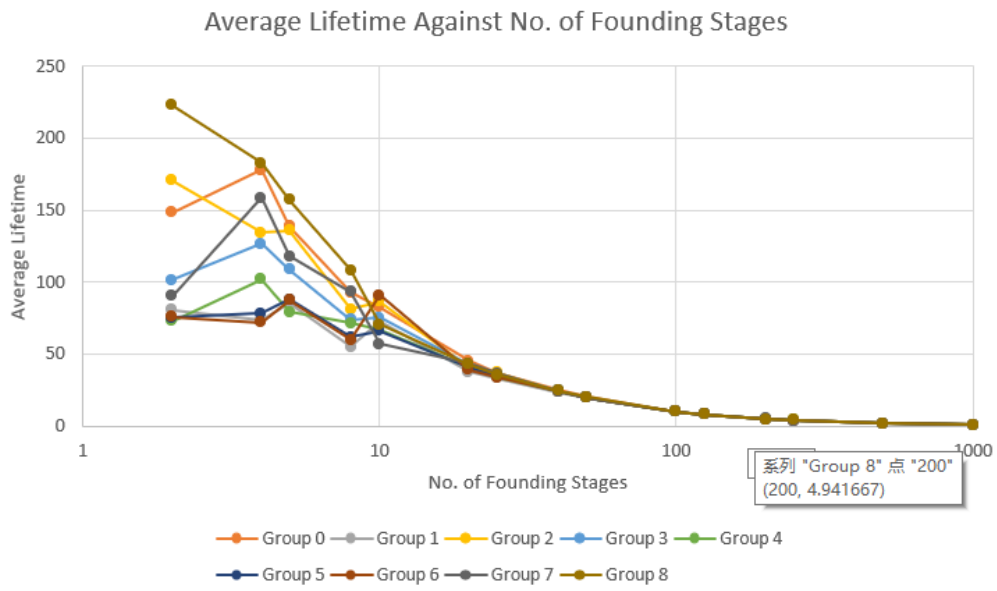
(d) Image 4



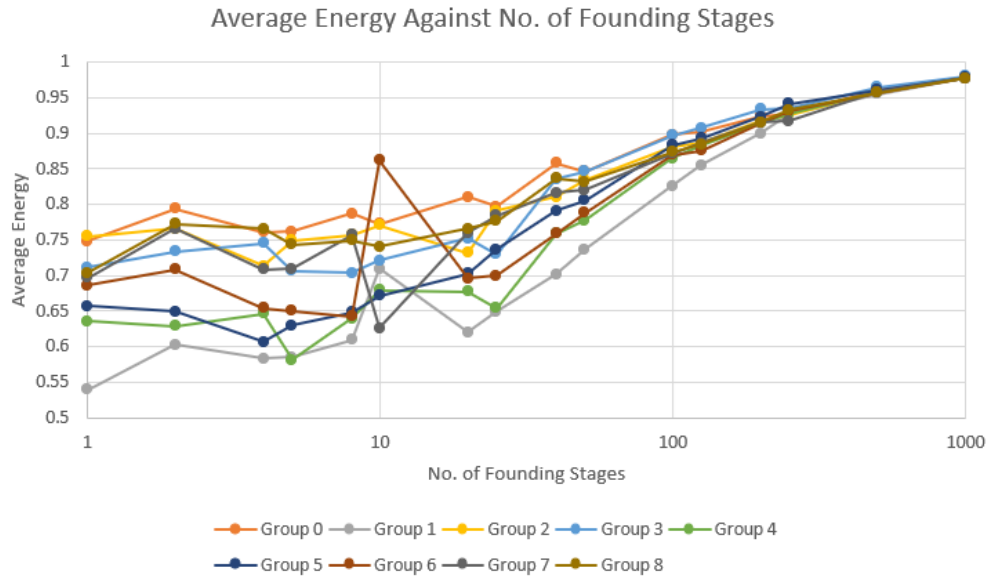
(e) Image 5



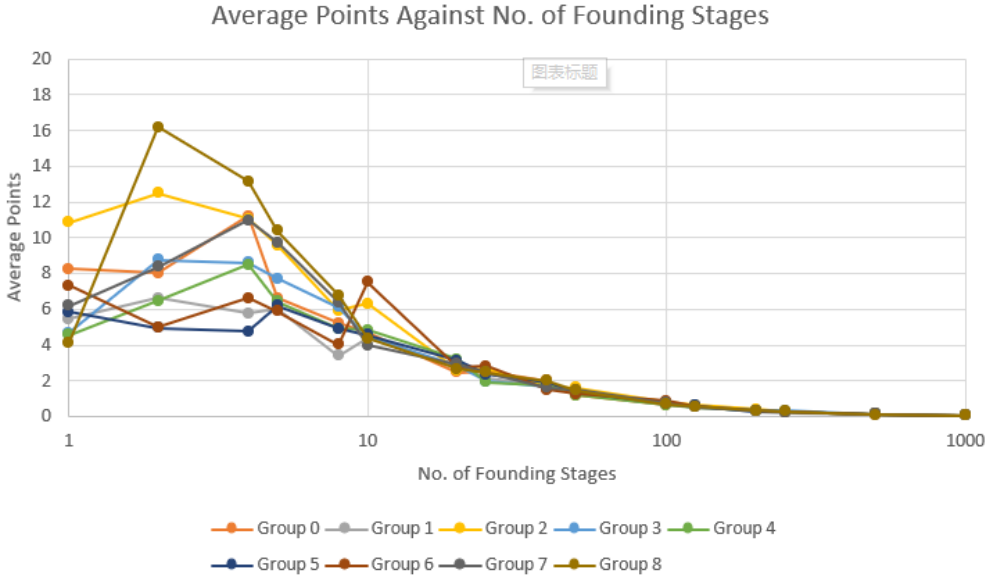
(f) Image 6



(a) Average Lifetime for agents from each teams



(b) Average Energy Level for agents from each teams



(c) Average Points for agents from each teams

Chapter 14: Experiment Analysis and Results for Group Agent Strategies

The purpose of this chapter is for individual agent groups to explore how well their strategies and agents performed in each experiment and environment. With each experiment and parameters conditions are changed. Group agents will therefore perform differently and experiments will give insight into how well each strategy works in different situations. This chapter will be sectioned by experiment and within each section there will subsection where groups will discuss how each strategy performed.

14.1 Experimenting with Governance: Deliberative Democracy

14.1.1 Group 1 Agent Strategy

As we can see in Figures 13.1 & 13.2, our agent performs better when it has free will than when it is forced to abide by the deliberative democracy governance. This is expected since our strategy takes into account reputation and relationships to choose which governance would give us the best chance at acquiring more points/energy. Our agent goes from performing in the bottom half of agents when forced into deliberative democracy, to the top half when having a free choice of governance. As the number of rounds increase, our agent progressively gets better and by round 10 has one of the highest life expectancies across all agents.

14.1.2 Group 2 Agent Strategy

For deliberative democracy, Agent 2 will vote for a lootbox which benefits them the most. If the voted lootbox has not been chosen through voting, Agent 2 will decide whether to conform based on the social capital of all the agents on the bike. If the agents on the bike have benefited Agent 2 in the past, there will be a greater chance that Agent 2 pedals towards the collectively chosen lootbox. If the bike has an overall low social capital, Agent 2 will continue to pedal towards a lootbox which has the highest reward. Even with the possibility of non-conformity on the bike, Agent 2 seems to yield good results within strictly democratic games, this can be attributed to the fact that most agents on the bike had a high social capital score thus increasing the probability of conformity.

14.1.3 Group 3 Agent Strategy

Our agent is committed to achieving a harmonious equilibrium between individual gains and collaborative efforts. This approach not only enhances the likelihood of accruing more points but also ensures the prolonged sustainability of our agent. The overall performance of our agent is deemed satisfactory in this regard.

14.1.4 Group 4 Agent Strategy

Our agent is still learning about the game and other agents. It is willing to sacrifice some lifetimes and energy to spend time learning in the beginning. This is why basebiker performs better in terms of lifetimes and energy average. But our agent is average in collecting points and tries to meet this goal even with limited lifetimes. But with more rounds the agent's performance will improve.

14.1.5 Group 5 Agent Strategy

In the comparative analysis of enforced democracy versus the normal state within the SOMAS World environment, the observations indicate a minimal difference in overall impact. When operating under an enforced democratic structure, agents displayed a marginal improvement in performance. Specifically, they demonstrated an average increase in longevity of 2.25 iterations, accompanied by a slight enhancement in points by 0.3 and a minor elevation in average energy levels by 0.1. This subtle improvement under the enforced democracy can be attributed to the agents' inherent strategic focus on democratic processes. The agents, predisposed to actively engage in democratic practices for esteem enhancement among peers, already exhibited behaviours conducive to a democratic environment. Consequently, the shift to an enforced democratic setting did not significantly alter their operational dynamics. This observation suggests that the agents' performance in SOMAS World is

closely tied to their strategic alignment with the governance model, highlighting the adaptability and resilience of their decision-making frameworks in varying political landscapes.

14.1.6 Group 6 Agent Strategy

Group 6 exhibited a consistent performance in terms of lifespan, without significant peaks or troughs compared to other groups. I believe our robust strategy ensured our solid survival in the Deliberative Democracy environment. Our method of continuously seeking Lootboxes of our target color ensured we didn't lose our stable progress due to short-term greed. At the same time, in terms of scoring, Group 6's results were quite average. Our proactive decision to leave the Mega-Bike when the energy level fell below a certain threshold might have missed some scoring opportunities, but it ensured our lasting survival, which is vital in the long run. Our preference for Mega-Bikes of the same color enhanced team collaboration, even if it meant we weren't always the highest scorers. Overall, while our strategy didn't keep us consistently at the top of the scoreboard, it endowed us with the ability to survive and adapt in the complex and dynamic world of SOMAS.

14.1.7 Group 7 Agent Strategy

Under an enforced democratic regime in SOMAS World, Agent 007 displayed a notable decline in performance. This downturn was characterised by a significant reduction in points accumulation and overall effectiveness when compared to other agents. This decrease suggests that the democratic environment may not align well with Agent 007's inherent operational strategies or decision-making frameworks. The performance of Agent 007 indicates a requirement for a stronger form of leadership to achieve success in the SOMAS World environment.

14.2 Experimenting with Governance: Leadership Democracy

14.2.1 Group 1 Agent Strategy

In this experiment, our agents exhibit relatively poor performance, as evident in the graphs (see 13.7). Our agent typically refrains from choosing leadership in a democracy unless it has formed very strong relationships with other agents. However, our agent is unexpectedly thrust into a leadership position without having established solid relationships with other agents. The BDI structure of our agent means that every action is governed by our beliefs which in this context are characterized by our relationships with other agents. Since these relationships have not yet been established, our agents perform poorly. Nevertheless, as the rounds progress our agent shows an improvement in retrospect to earlier rounds which highlights our agent's ability to develop beliefs as it continues to interact with other agents and form its opinion.

14.2.2 Group 2 Agent Strategy

Agent 2's strategy with leadership depends on the social capital of each agent on the bike. Since each agent on the bike is assigned a social capital score once they have encountered Agent 2 (been on the same bike), this social capital score is used to apply weighting to respective agents. This incorporates the concept that agents who have benefited Agent 2 will be rewarded with a heavier weight for their vote. When voting for a leader, Agent 2 will always vote for themselves and the agent that has had the highest social capital within the game round. If the agent with the highest social capital is not on the same bike as Agent 2, the votes for all other agents will be 0, whilst the votes for themselves will be 0.5. This strategy allows Agent 2 to gain better game statistics as they choose to elect and give deciding power to agents who have benefited them in the past.

14.2.3 Group 3 Agent Strategy

Our agents had a high lifespan and points as shown in the figures. The energy level was not high which meant our agent was willing to pedal in the leadership setting. The high points showed that the leader recognized the efforts and shared the points. The long lifespan showed the stability and the reliability of our agent performance in the leadership setting.

14.2.4 Group 4 Agent Strategy

Our agent performs better than the BaseBiker on an average when comparing lifetimes and points accumulated but not for the energy. Here, the reputation and honesty matrix values help the agent perform better but have not had as much of an impact as expected for energy values.

14.2.5 Group 5 Agent Strategy

From the Leadership Democracy experiment, it can be clearly seen that team 5's agent performs the best in terms of longevity compared to itself in other governing systems as can be seen in the figure below. This is likely due to the fact that the agent is able to capatilise on the esteem it has built from past interactions it has had with others.

14.2.6 Group 6 Agent Strategy

Our team's leadership performance was slightly inferior compared to when we had free choice. This is because our agent design inherently focuses more on governance in autocratic and democratic scenarios, and thus did not effectively distribute leadership roles in our expulsion strategy. Nevertheless, thanks to our excellent reputation mechanism, we were still able to achieve a general benchmark level.

14.2.7 Group 7 Agent Strategy

In this experiment, our agent demonstrated improved performance under a leadership-driven democracy when compared to a full democracy model. While the performance under leadership democracy was not outstanding, it was notably better than in a fully democratic setting. This observation suggests that our agent is more suited to scenarios where a leader is present to align decision-making processes.

14.3 Experimenting with Governance: Dictatorship

14.3.1 Group 1 Agent Strategy

In this experiment, our agent performs slightly below average as indicated by the resulting graphs (see 13.12). The strategy employed by our agent involves opting for a dictatorship only when our relationships with others are exceptionally strong, increasing the likelihood of being selected as the dictator. However, again our agent is compelled into a dictatorship unexpectedly. Being primarily a relationship-centric BDI agent, our agent bases most of its actions on its beliefs, particularly the relationships it has with other agents. In our case, these relationships are not well-established during the early rounds, causing our agent to under perform, aligning with a lack of information about fellow bikers. As the rounds progress, our agent shows a slight improvement in performance corresponding with clearer strategic intent based on the additional biker information. Since only the dictator's fairness score is affected (as they are distributing the loot), our overall opinion of all agents not fully updated all the time, hence the slower learning rate. Our agent's dictator-based decision making is reinforced with clarity in our relationships.

14.3.2 Group 2 Agent Strategy

Dictator voting is the basis for Agent 2's dictatorship strategy. When voting for a dictator, Agent 2 will vote for themselves and the agent with the historically highest social capital score. If the agent with the highest score isn't present on the current bike, all other agents will receive 0 votes. If Agent 2 is elected as the dictator, the desired lootbox will be one with the most reward within pedalling distance. Agent 2 also actively avoids the Awdi, if an Awdi is near, Agent 2 will redirect to a lootbox which is the nearest lootbox away from the Awdi. By dictating the bike to pedal towards lootboxes with the highest reward, Agent 2 can last much longer within the game, which can be observed by the long life expectancy.

14.3.3 Group 3 Agent Strategy

Under the enforced dictatorship, Group 3 agents exhibit a life cycle marked by high performance, though it's tinged with occasional instabilities. Remarkably, their energy levels average higher than all other groups, maintaining a remarkable degree of stability. However, when it comes to scoring points, Group 3's performance is characterized by significant fluctuations, placing them squarely in the middle among the other groups.

14.3.4 Group 4 Agent Strategy

Our group performs well in dictatorship and it has a better performance than the BaseBiker in lifetime and points. However, we observe a shortfall in energy performance. This could be attributed to our agent's tendency to trust others in the initial stages without having fully developed the sophistication required within the short span of 10 rounds.

14.3.5 Group 5 Agent Strategy

In the SOMAS World simulations, the agent's dictatorship approach, blending centralized decision-making (DictateDirection) and meritocratic resource allocation (DecideDictatorAllocation), resulted in favourable outcomes:

Enforced Dictatorship Lifetime (Avg: 96.94)

The top-down governance style likely contributed to this high average, indicating sustained control and strategic adaptability in leadership. With an average of 96.94, the agent had high longevity in most simulations. This suggests the strategy was effective in maintaining power, possibly due to effective resource management.

Energy Average (Avg: 0.902283)

This reflects stable resource levels, suggesting effective utilisation and allocation, aligning with the meritocratic principles of DecideDictatorAllocation. The energy average of 0.902283 indicates a relatively stable resource level. This could be a result of efficient energy usage or successful energy acquisition strategies.

Point Average (Avg: 4.802923)

Moderate success here may reflect the balance between resource management and other objectives, underpinned by a strategic focus on reputation and social norms. An average of 4.802923 points indicates moderate success in achieving objectives. This might reflect a balance between maintaining power and other objectives like resource collection or alliance formation.

Conclusion

The agent's performance in maintaining power and resource levels could be attributed to the strategic approach outlined in the report. Concepts like reciprocal altruism and social norm compliance likely contributed to the agent's ability to sustain its position and manage resources effectively. The strategy's emphasis on the Economy of Esteem might have facilitated beneficial interactions with other agents, aiding in longevity and resource management. The overall success reflects a well-implemented strategy that balances power maintenance with resource optimization.

14.3.6 Group 6 Agent Strategy

These charts reflecting the enforced dictatorship scenario showcase the strength and astute strategic planning our team prides itself on. Our consistent life expectancy speaks volumes of our adaptability and the robust survival tactics we've honed in a system that demands alertness and agility under a centralized command. Our energy management, a testament to our disciplined approach, has kept us from extremes, ensuring a steady supply to support our endeavors.
Our scoring average is not just a number—it's a narrative of our tenacious pursuit, a testament to our cohesive team dynamics, and a reflection of our innovative scoring system that incentivizes unity and collective ingenuity. It's through our unique strategies—like the relentless pursuit of Lootboxes and the tactical alignment with like-colored Mega-Bikes that we have carved out a path of resilience. Particularly, our 'forgiveness' policy has not only cultivated a spirit of solidarity but has also enhanced our team's morale, underscoring the success that comes from compassionate teamwork.

14.3.7 Group 7 Agent Strategy

When a dictatorship was enforced, Agent 007's performance saw a marked improvement. The agent moved to a mid-tier position in terms of points accumulation, aligning more closely with the average performance of other agents. This shift indicates that Agent 007's strategic approaches are more suited to a centralised, authoritarian governance model when compared with any other governance structure.

14.4 Experimenting with Governance: Choosing Governance Strategy

14.4.1 Group 1 Agent Strategy

Across the three resulting graphs (see Figure 13.16), our agents exhibit relatively average performance. One notable observation is the improvement in our agent's performance towards the end of the experiment. This positive trend is evident in all graphs: the energy graphs show an increase from an initial average of 0.65 to a final average of 0.88; the lifetime graph depicts a rise from an initial average of 70 to a final average of 80, and

in the average points graph, we observe an ascent from around 4 to 7. These findings suggest that our agents perform better with greater clarity of it's relationships with others, a result of iterated update in opinions based on our BDI feedback loops. Furthermore, it shows that our agent strategy works as intended, since the primary idea behind it was creating a social agent.

14.4.2 Group 2 Agent Strategy

Typically our agent follow cohort trends in regards to performance in governance systems, and this freedom is no exception. Notably, despite other systems potentially having higher yields, our agent is averse to dictatorship and prefers democracy, so in the case of freedom of choice, will opt towards democratic systems, leading to a performance most similar to democracy, even with the additional voting costs for governance system impacting our performance.

14.4.3 Group 4 Agent Strategy

In this experiment, the performance of the agent of group 4 of is relatively good. Its performance is not easily influenced by external factors, which shows a relatively strong robustness. The agents in the group 4 exhibit strong ability to recover from mistakes. For instance, based on the average lifetime data, the value dropped to only 48 in the 8th round, but by the 9th round, it recovered to approximately 70. This indicates that the agents in the group 4 possess strong learning capabilities. With a sufficiently larger number of rounds in the game, theoretically, such agents could achieve even better performance.

14.4.4 Group 5 Agent Strategy

The graph below illustrates the agent's lifespan in different governance schemes from earlier experiments. Notably, the agent performs most effectively in leadership roles but less so in democratic setups.

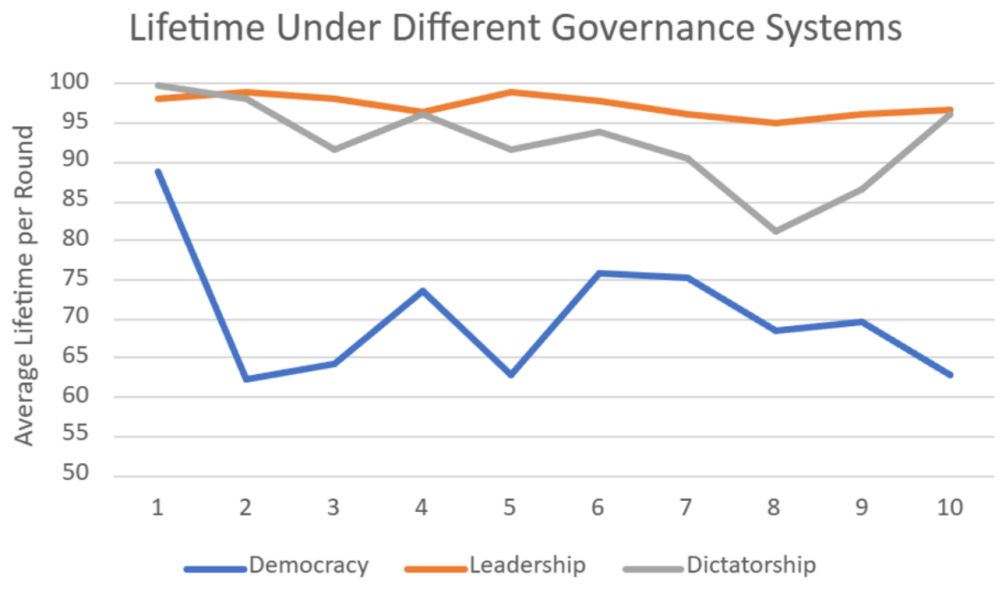


Figure 14.1: Overall Comparison-Leadership vs Free choice

However, the agent's lifespan is significantly shorter when it has autonomy in decision-making, as opposed to situations where it is compelled to be in leadership roles or interact with leaders. However, this independent lifespan still outperforms that observed in a purely democratic environment. This pattern is attributed to the agent's behavior, where it initially opts for a democratic setting to enhance its esteem but eventually defaults to favoring leadership opportunities in areas crucial for its survival.

14.4.5 Group 6 Agent Strategy

Our previous approach centered on dictatorship, but we change our strategy to leadership or democracy if our objectives don't align with the majority on the bike. We notice an average lifetime but significantly higher scores in average points gained. The potential reasoning behind this lies in the dynamics: when an agent opts for dictatorship, the dictator compels all agents to pedal towards the target. Conversely, in cases of leadership or

democracy, there might be reluctance to pedal when the bikes steer towards an unfavorable direction, resulting in lower average scores. This explains why our agents under dictatorship tend to obtain higher points than others.

14.4.6 Group 7 Agent Strategy

In the SOMAS World simulation, where agents had the option to select their own leadership strategies, Agent 007 always chose a deliberative democracy approach. Initially, this strategy was presumed to be optimum for Agent 007. However, a retrospective analysis of Agent 007's performance across various experiments indicated that this preference was not the optimal choice, but a choice that resulted in the worst performance among the tested strategies. This was why Agent 007 performed so poorly in instances where they could choose the governance structure.

14.5 Experimenting with Changing Threats: Energy Scarcity

14.5.1 Group 1 Agent Strategy

As the number of lootboxes per agent increases, our average points and average lifetime increase as can be seen in graphs 13.18b and 13.18a owing to greater opportunities to obtain more energy and points. Overall, our agent is in the top half of performance in both categories when comparing with other team's agents showing that it is able to adapt to different situations.

14.5.2 Group 2 Agent Strategy

When lootboxes become scarce, Agent 2 will naturally choose depending on the governance for the specific bike. As a dictator, Agent 2 will still choose to pedal towards a box which has the highest gain within a reasonable distance. As a leader, Agent 2, will choose a lootbox which has been voted for those who have benefited them in the past. In deliberative democracy, Agent 2 will have a probability of conformity, where it decides whether to pedal in the decided direction depending on the overall social capital on the bike. All of these strategies don't necessarily account for the scarcity of lootboxes, therefore, experiment results show Agent 2 doesn't necessarily excel in these games.

14.5.3 Group 3 Agent Strategy

Overall, team3 has performed a good job regarding the quantity of Lootboxes considering possible fluctuations. Our agents showed a greater inclination towards collaboration, which is evident from the high scores achieved when the Lootboxes collected per agent=4.

14.5.4 Group 4 Agent Strategy

In general, Team4 has done relatively well in all quantities of Lootboxes, showing remarkable stability across various numbers of Lootboxes. Even in scenarios with fewer reward boxes, they consistently achieve favourable results.

14.5.5 Group 5 Agent Strategy

Examining the experiment data provided, the lifetime and energy of team 5's agent decrease with a reduction in the number of Lootboxes available. It is important to note that this is a trend seen with most other agent strategies and is an expected outcome. However, there is substantial evidence suggesting the effectiveness of the state machine integrated into team 5's strategy, as the energy changes at a notably reduced rate when the Lootbox count decreases.

For instance, when the Lootbox count decreases from 2 boxes available per agent to 1 box available per agent, in comparison to the BaseBiker implementation, Team 5's agent experiences only approximately a 5% decrease in energy, whereas the BaseBiker's energy decreases by 10%. This distinction clearly indicates the active role of the state machine in mitigating energy loss.

14.5.6 Group 6 Agent Strategy

Under the governance of a dictatorship within Team 6, our primary focus remains on ensuring a sustained survival rate across various scenarios involving different quantities of Lootboxes. Our dictator's core strategy revolves around preserving the lives of all team agents while maximising points for the dictator. As Lootbox distribution becomes denser, our points increase due to the augmented resource availability, consistent with

the overall trend. However, we encounter significant variability in both the distribution of lifetime and points acquired as the agent chosen as the dictator tends to gain considerably more points than those being as normal bikers.

14.5.7 Group 7 Agent Strategy

It has been observed that Agent 007 maintains a higher-than-average pedalling rate when compared with other agents. This leads to increased energy consumption which becomes particularly impactful in scenarios where Lootboxes are scarce. In such a game of scarcity, Agent 007 struggles to maintain its energy levels due to its high energy usage, which adversely affects its overall performance.

Conversely, in environments where Lootboxes are more prevalent, the increased availability of resources allows the agent to sustain its high energy consumption without detrimental effects to its lifetime. Moreover, its ability to move rapidly is likely to enhance its effectiveness in acquiring more points, as it can reach and utilise Lootboxes more efficiently than agents with lower pedalling rates. Therefore, the fluctuating availability of Lootboxes in SOMAS World plays a critical role in determining the performance outcomes for Agent 007.

14.6 Experimenting with Founding Stages: Varying Frequency of Founding Stages

14.6.1 Group 1 Agent Strategy

In this experiment our agent performs relatively poorly. With one founding stage our social agent is not able to base it's decision of governance on solidified friendships since it has not yet formed opinions about other agents. As the number of founding stages increases, our agent is able to slowly improve it's performance with more reliable knowledge on it's relationship with others, as can be seen in Figure 13.20b.

14.6.2 Group 2 Agent Strategy

Agent 2 implementation of Social Capital ensures that a map for all other agents it has encountered between various iterations and rounds is preserved and updated accordingly. This map serves as the medium through which experience/knowledge about other agents is carried throughout the game. Although most agent groups seem to converge (in all graphs) once 100 founding stages are reached, Agent 2 appears to follow a similar trend that the larger the number of iterations in a given round, the larger its average lifetime and points. Alternatively, the lower the number of founding stages, the more time Agent 2 appears to have lasted and accumulated points. This meant that given a survival scenario where there were few, if any, founding stages (where energy levels were refilled), Agent 2 would coordinate/make better decisions with its fellow bikers to survive the longest compared to most of the other agents.

Concerning the average energy, it makes sense that the shorter a round became, the higher the average energy level became as all the agents expended through pedalling, spending time in the void and voting. However, Agent 2 appears to have beaten the other agents when the founding stages were minimal, meaning that Agent 2 faired better when confronted with a longer game with 1000 iterations, similar to before when talking about lifetime and points.

14.6.3 Group 3 Agent Strategy

In the experiment involving varying leadership change frequencies, our agent performed best in stages with gradual increases and averaged in other scenarios. We speculate this might be because our strategy focused on emulating others, avoiding standing out excessively or lagging behind, leading us to remain in the middle of the pack.

14.6.4 Group 4 Agent Strategy

In the experiment about varying frequencies of leadership changes, agents in group4 demonstrated satisfactory performance in the initial stages and performed well in the mid-to-late stages. This is probably because our agents tend to initially trust other agents and subsequently assess reputations based on their behaviours. This dynamic necessitates a learning process for the agents.

14.6.5 Group 5 Agent Strategy

The outcome of this experimentation is not directly affected by the agent's strategy and the only results are due to the dynamics that arise from changing the frequency of founding stages. For example, as the frequency

increases the average energy tends towards 1 as it is reset to this in each founding stage. Similarly, the average lifespan and points drop to 0 as each bike has its location reset at each founding stage making it impossible for any bike to reach a Lootbox for points.

14.6.6 Group 6 Agent Strategy

As the number of founding stages rises, our agent's performance notably deteriorates due to insufficient information available within limited iterations. When the number of founding stages is appropriately chosen (around 5-10), our agent demonstrates an average performance level.

14.6.7 Group 7 Agent Strategy

As discussed, our agent had a strategy of consistently voting for deliberative democracy and therefore did not adapt its strategy at each founding stage. Despite this, the agent's performance was consistently middle tier in all metrics before its performance converged with other agents as the number of founding stages increased beyond 100. This suggests that the governance strategy of the agent did not put it at any disadvantage as the number of founding stages were changed.

14.7 Experimenting with Knowledge Management: Varying ACL

14.7.1 Group 1 Agent Strategy

As we can see in Figures 13.23 & 13.26, our agent performs better with messaging enabled. This is expected from the BDI structure of our agent as the more accurate our beliefs are, the more reliably our intentions are justified. Introducing messaging to the environment brings more information to refine our relationship calculations, such as clarity behind effort and direct experience within our trust framework, particularly when in a position of leadership where decisions are impacted more by opinions. In addition, our agent only takes into account messages from trustworthy agents minimizing any harm to our agent's performance.

14.7.2 Group 2 Agent Strategy

Many groups saw improved performance with an expanded ACL, our agent benefitted minimally with lower points with the ACL but higher lifetime. Most notably, we stayed fairly consistent with or without messaging, suggesting our implementation was robust to handle the advantages and disadvantages of messaging, those being increase knowledge of agents and the environment with messaging, but other agents having greater knowledge of us, which may cause harsher sanctions if we are trying to contribute less to the bike.

14.7.3 Group 4 Agent Strategy

Messaging is a key component for our agent. It is used to calculate the honesty value of all agents and build an honesty matrix. Our agent compares messages received to the actions to perceive how honest an agent is. Thus, when messages are enabled, it has a better way of evaluating other agents. This helps our agent make a more optimal decision which leads to longer survival in the game.

14.7.4 Group 5 Agent Strategy

When the messaging system is removed, the ability of the agent to convey intention and praise is restricted. This reduces the agent's ability to work cooperatively and reduces social coordination across the board.

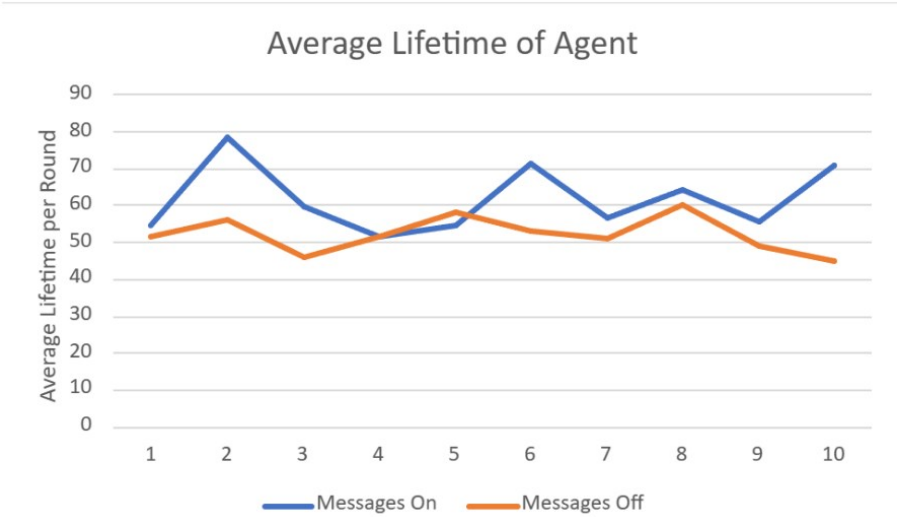


Figure 14.2: Average Life Expectancy of Agents across 10 simulations with messaging system enabled and disabled.

This difference in performance is clearly illustrated in Figure x, where the lifespan of the agent drops from an average of 61.89 down to 52.12 when the messaging is removed. The reduced lifespan indicates a heightened vulnerability, potentially resulting from a lack of coordinated efforts and shared strategies.

As well as the average points dropping from 3.36 to 0.61. These results strongly support the premise that a lack of communication directly impacts social cohesion among the agents. The inability to convey intentions and offer praise hampers the development of cooperative strategies, ultimately leading to a suboptimal performance.

It is possible that team 5's agents' increased efforts are not being recognised by the fellow bikers, since without the messaging system, they cannot know the extent to which the agent is selflessly trying to help everyone in succeeding. Furthermore, messaging is a tool in helping the agent to more easily determine which biker is not exerting enough effort, lacking or purposefully idling, which makes it even harder for team 5's agent to adjust their strategy accordingly.

Overall, as team 5's agent main strategy is based on The Economy of Esteem, removing the messaging system hampers the ability for the agent's esteemed reputation to propagate through SOMAS World, therefore reducing the impact of the agent strategy.

14.7.5 Group 6 Agent Strategy

Before and after the addition of the messaging system, Agent 6 maintained high performance, benefiting from our trust strategy and robust messaging infrastructure. After the introduction of messaging, our lifetime ranked among the top two, and our average score and energy acquisition were significantly higher than other groups. The introduction of the messaging system maximized the effectiveness of our trust system. It's evident from the overall increase in survival time, energy acquisition, and scoring that by communicating with other groups through messages, the overall survival rate of the societal environment improved, undoubtedly proving the great success of our group's messaging system.

14.7.6 Group 7 Agent Strategy

Messaging: Agent 007's performance was not significantly influenced by the messaging feature's status. The results highlight Agent 007's consistent placement in the lower tier of the rankings, regardless of whether messaging was enabled.

A notable aspect of the results is the greater variance in Agent 007's point distribution during the absence of messaging, a pattern that was not unique to this agent alone. A potential explanation for this observation could be the lack of communication, leading agents to disregard the Lootbox choices of others. Consequently, agents may have been more inclined to leave or stay on bikes with agents of the same colour, driven by diminished trust in agents deemed untrustworthy because they continued to vote for different Lootboxes. Evidence for this was shown using the visualiser when messaging was disabled, see Figure 14.3. This would result in some bikes agreeing more on what Lootbox to head to, reducing disputes and divisive actions from other agents. This

would lead to agents gaining more points if they found agents of their colour. Conversely, the inability to locate a compatible group could result in reduced point gains for that round.

Figure 14.3: Visualiser Screenshots Showing Agents Grouping by Colour When Messaging is Disabled

The results of the experiment, however, do not give sufficient evidence to reliably conclude this hypothesis. Further trials seeing this behaviour repeated would lead to a more reliable conclusion of this. Furthermore, an enhancement of the messaging system to require agent acknowledgement and response could lead to more insightful outcomes. An in-depth analysis of how agents process and act upon messages could shed light on their interactive behaviours and the impact of disabling messages.

14.8 Managing Relationships with Difficult Members: Adding Narcissist Agents

14.8.1 Group 1 Agent Strategy

It can be noted that our agent performs relatively the same when either an NPC narcissist agent exists or not. This can be merited to our agent opinion formation enforcing penalties on the narcissistic agent and disregarding it and aiming to build better relationships elsewhere. The results also indicate a greater range in our agent's lifetime and points when a narcissist is included. This can be potentially characterised by an interaction with a narcissistic agent and our agent's initial acceptance of such agent (since it hasn't fully developed an opinion), resulting in our agent having an unsuccessful round.

14.8.2 Group 2 Agent Strategy

In this case the impact on our agent is minimal. While we perform better in a world without narcissists, the difference is minimal which is somewhat unsurprising since our favour implementation and limited forgiveness allows us to identify agents acting against group interests and vote to remove them swiftly, while if we are taking a less collaborative approach the addition of narcissists may provide a smoke screen that hides our true intent, although with a slightly worse performance with narcissists the benefit of narcissists seems very minor.

14.8.3 Group 4 Agent Strategy

The agent performed well in this test. It shows a great stability.The agents can maintain good performance even after being disturbed by narcissists.Moreover, after multiple rounds of experiments, the agent exhibited a trend of continuous learning. It can make optimal decisions better if it understands the narcissist agent's intentions through reputation and honesty matrix values. It will perform better if the number of rounds increase as it does take some time to learn about all the agents initially.

14.8.4 Group 5 Agent Strategy

As the agent strategy is built around reciprocated generosity and garnering an economy of esteem, it is logical to assume that the addition of narcissistic agents will harm the agent. Due to this assumption fallbacks were implemented to ensure that the agent wouldn't be taken advantage of. Systems such as the survival state and trust decay allows team 5's agent to mitigate its generosity being exploited by other agents.

Although these systems were put in place, it seems that more rigorous safety measures should have been included. When narcissistic agents are added to the world, the agent's performance decreases. Its average lifetime decreases by 11.58 iterations. This indicates a heightened vulnerability of the agent in the presence of narcissistic counterparts. The shorter lifespan not only compromises the agent's ability to survive but also impacts its capacity to gain points and contribute to a more successful overall performance. This also results

in lower average points gained overall, dropping from 4.67 to 3.4. This reduction in points suggests that the agent's generosity and cooperative strategies are less effective in the presence of agents driven by self-centred motives, emphasising the need for more comprehensive and adaptive countermeasures.

14.8.5 Group 6 Agent Strategy

Our agent shows strong performance throughout the experiment, both in terms of average points and average lifetime. As our agent is characterized by a sophisticated use of social capital in decision-making, it enables the agents to navigate the Narcissist Agents experiment effectively by building and maintaining strong cooperative relationships, adapting to changes, and strategically allocating resources based on trust. . This holistic and dynamic approach be the reason behind their superior performance in the Nacrissist Agents experiment.

14.8.6 Group 7 Agent Strategy

An attempt to incorporate a narcissistic agent into the SOMAS World was made using Agent 007 personality framework. Nonetheless, this alteration had a negligible effect on performance when compared with the regular Agent 007. This indicates that the broader strategic approaches of Agent 007 overshadowed personality adjustments. For a comprehensive evaluation, a greater number of iterations would be required as well as more experimentation with the narcissistic traits.

Chapter 15: Conclusion

As we say our final good-byes to SOMAS World, we walk away better software engineers, not only for the coding skills required for creating this project, but also for the higher-level understanding we have gained about the ways that our global society can be understood and modelled by machines.

Within our attempt at doing so, we gained interesting insights: in SOMAS World, we saw that self-interested agents outperformed cooperative agents. Thus, SOMAS World suffers from the Tragedy of the Commons. This was due to the limitations of SOMAS World as we were able to design it in our limited timeframe; future work on SOMAS World would need to focus on increasing system transparency and more justice-centric parameters.

As a semi-self-organised cohort of 2023, we are grateful that we do not live in a real world made up of Awdis and Mega-Bikes, and equally grateful for the adventure of a project this has been. Our undying thanks goes out to the wonderful GTAs for their guidance and our professor Jeremy Pitt for his instruction and passion for multi-agent systems.

Bibliography

[1] T.K. Ahn and Elinor Ostrom. "A Conceptual Framework for the Study of Social Capital in New Cooperative Institutions". In: *Indiana University Workshop in Political Theory and Policy Analysis* (2001).

[2] Jones et al. "Personality, Emotions and Physiology in a BDI Agent Architecture: The PEP - BDI Model". In: *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*. Vol. 2. 2009, pp. 263–266.

[3] Marshall et al. "Validation of the five-factor model of personality across instruments and observers". In: *Journal of Personality and Social Psychology* 67 (1994), pp. 278–286.

[4] Zoumpoulaki et al. "A Multi-agent Simulation Framework for Emergency Evacuations Incorporating Personality and Emotions". In: *Artificial Intelligence: Theories, Models and Applications*. Springer, 2010, pp. 423–428.

[5] Robert Axelrod. "More effective choice in the prisoner's dilemma". In: *Journal of conflict resolution* 24.3 (1980), pp. 379–403.

[6] Peter L. Berger and Thomas Luckmann. *The Social Construction of Reality*. 1966. URL: https://web.archive.org/web/20191009202613id_/http://perflensburg.se/Berger%20social-construction-of-reality.pdf.

[7] Philip J Boland. "Majority systems and the Condorcet jury theorem". In: *Journal of the Royal Statistical Society Series D: The Statistician* 38.3 (1989), pp. 181–189.

[8] Pierre Bourdieu and Loïc Wacquant. "Symbolic capital and social classes". In: *Journal of classical sociology* 13.2 (2013), pp. 292–302.

[9] Geoffrey Brennan and Philip Pettit. *The economy of esteem: An essay on civil and political society*. OUP Oxford, 2004.

[10] Geoffrey Brennan and Philip Pettit. *The Economy of Esteem: An Essay on Civil and Political Society*. Available from: https://www.amazon.co.uk/Economy-Esteem-Essay-Political-Society/dp/0199289816, [Accessed 13 December 2023]. 2005.

[11] *BRIA 23 3 c Justice as Fairness: John Rawls and His Theory of Justice*. 2007. URL: https://www.crf-usa.org/bill-of-rights-in-action/bria-23-3-c-justice-as-fairness-john-rawls-and-his-theory-of-justice#:~:text=The%20most%20controversial%20part%20of,go%20to%20the%20least%20advantaged. (visited on 12/12/2023).

[12] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. "Multi-agent reinforcement learning: An overview". In: *Innovations in multi-agent systems and applications-1* (2010), pp. 183–221.

[13] Cristiano Castelfranchi et al. "Personality traits and social attitudes in multiagent cooperation". In: *Applied Artificial Intelligence* 12.7-8 (1998), pp. 649–675.

[14] Amrita Chakraborty and Arpan Kumar Kar. "Swarm intelligence: A review of algorithms". In: *Nature-inspired computing and optimization: Theory and applications* (2017), pp. 475–494.

[15] Nicolas de Condorcet. *Essay on the Application of Analysis to the Probability of Majority Decisions*. Cambridge University Press, 2014. DOI: 10.1017/CBO9781139923972.

[16] Daniel C. Dennett. *The Intentional Stance*. Available from: https://mitpress.mit.edu/9780262540537/the-intentional-stance/, [Accessed 12 December 2023]. 1989.

[17] Guy Theraulaz Eric Bonabeau Marco Dorigo. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford: Oxford University Press, 1999. DOI: 10.1093/9780195131581.001.0001.

[18] R. M.L. Evans. "Pay-off scarcity causes evolution of risk-aversion and extreme altruism". In: *Scientific Reports* 8.1 (2018). ISSN: 20452322. DOI: 10.1038/s41598-018-34384-w.

[19] Adam Green. "Forgiveness and the Repairing of Epistemic Trust". In: *Episteme* (2021), pp. 1–17.

[20] Jerald Greenberg, Claire Elizabeth Ashton-James, and Neal M. Ashkanasy. "Social comparison processes in organizations". In: *Organizational Behavior and Human Decision Processes* 102 (2007), pp. 22–41. URL: https://api.semanticscholar.org/CorpusID:62823567.

[21] Garrett Hardin. "Tragedy of the Commons". In: (1968). Available from: https://www.econlib.org/library/Enc/TragedyoftheCommons.html, [Accessed 12 December 2023].

[22] R. Hegselmann and U. Krause. "Opinion dynamics and bounded confidence: Models, analysis and simulation". In: *Journal of Artifical Societies and Social Simulation (JASSS) vol.5, no. 3* (2002). URL: https://www.jasss.org/5/3/2/2.pdf.

[23] R Hegselmann and U Krause. "Opninion dynamics and bounded confidence models". In: *Analysis and Simulation, JASSS* 5.3 (2002).

[24] Rainer Hegselmann and Ulrich Krause. "Opinion dynamics and bounded confidence: models, analysis and simulation". In: *J. Artif. Soc. Soc. Simul.* 5 (2002). URL: https://api.semanticscholar.org/CorpusID:8130429.

[25] Carl Hewitt. "Offices Are Open Systems". In: *ACM Transactions on Information Systems (TOIS)* 4.3 (1986). ISSN: 15582868. DOI: 10.1145/214427.214432.

[26] Francis Heylighen. "Stigmergy as a universal coordination mechanism I: Definition and components". In: *Cognitive Systems Research* 38 (2016), pp. 4–13.

[27] Geoffrey M. Hodgson. "Evolutionary Economics, in Fundamental Economics." In: *Evolutionary Economics* (2023).

[28] H. S. Houthakker and Kenneth J. Arrow. "Social Choice and Individual Values." In: *The Economic Journal* 62.246 (June 1952), p. 355. ISSN: 00130133. DOI: 10.2307/2227013.

[29] "International encyclopedia of the social and behavioral sciences". In: *Choice Reviews Online* 53.03 (2015). ISSN: 0009-4978. DOI: 10.5860/choice.192822.

[30] Pitt Jeremy et al. "Self-Organising Common-Pool Resource Allocation and Canons of Distributive Justice". In: *2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems*. 2012, pp. 119–128. DOI: 10.1109/SASO.2012.31.

[31] James Kennedy. "Swarm intelligence". In: (2006), pp. 187–219.

[32] Jong-Hwan Kim, Chi-Ho Lee, and Kang-Hee Lee. "Evolutionary generative process for an artificial creature's personality". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 39.3 (2009), pp. 331–342.

[33] Walter Krämer. "Kahneman, D. (2011): Thinking, Fast and Slow". In: *Statistical Papers* 55 (Aug. 2014). DOI: 10.1007/s00362-013-0533-y.

[34] Zachary F Lansdowne and Beverly S Woodward. "Applying the Borda ranking method". In: *Air Force Journal of Logistics* 20.2 (1996), pp. 27–29.

[35] Dan Lawrence. *Documentation - Home Page — Pygame GUI 0.6.9 documentation*. https://pygame-gui.readthedocs.io/en/latest/. Accessed: 2023-12-13. 2019.

[36] Witkowski M., Artikis A., and Pitt J. "Experiments in Building Experiential Trust in a Society of Objective-Trust Based Agents". In: *Trust in Cyber-societies* 2246 (2001), 111—132. DOI: 10.1007/3-540-45547-7_7.

[37] Busquets D. Macbeth S. and J Pitt. "System modeling: Principled operationalization of social systems using Presage2". In: (2014).

[38] A. H. Maslow. "A theory of human motivation." In: *Psychological Review* (1943).

[39] A. H. Maslow. "A theory of human motivation". In: *Psychological Review* 50.4 (1943). ISSN: 0033295X. DOI: 10.1037/h0054346.

[40] Robert R McCrae and Paul T Costa. "The five-factor model of personality as a framework for personality-health research". In: *Journal of personality and social psychology* 52 (1987), pp. 81–90.

[41] Michael E. McCullough. "Forgiveness: Who does it and how do they do it?" In: *Current Directions in Psychological Science* 10.6 (2001). ISSN: 09637214. DOI: 10.1111/1467-8721.00147.

[42] Shelley McKeown, Reeshma Haji, and Neil Ferguson. *Understanding Peace and Conflict Through Social Identity Theory: Contemporary Global Perspectives*. Springer International Publishing, 2016. DOI: 978-3-319-29869-6.

[43] Merriam-Webster. *Idiot - Etymology*. Merriam-Webster.com Dictionary. Available from: https://www.merriam-webster.com/dictionary/idiot, [Accessed 12 December 2023]. 2023.

[44] A. Mertzani et al. "Expertise, Social Influence, and Knowledge Aggregation in Distributed Information Processing". In: *ARTIFICIAL LIFE* 29 (2023), pp. 37–65. DOI: 10.1162/artl_a_00387. URL: https://doi.org/10.1162/artl_a_00387.

[45] Katarina Milanovic and Jeremy Pitt. "The Social Construction of "Shared Reality" in Socio-Technical Systems". In: *Trust Management XII. (IFIPTM). IFIP Advances in Information and Communication Technology*. Vol. 528. Berlin, Heidelberg: Springer, 2018, pp. 149–159.

[46] Nils J. Nilsson. "Teleo-Reactive Programs for Agent Control". In: *CoRR* cs.AI/9401101 (1994). URL: https://arxiv.org/abs/cs/9401101.

[47] Josiah Ober. *Demopolis: Democracy Before Liberalism in Theory andPractice*. 2017. URL: https://www.law.berkeley.edu/wp-content/uploads/2018/01/Demopolis_Democracy-Before-Liberalism-in-Theory-and-Practice.pdf.

[48] E. Ostrom. "Governing the commons: the evolution of institutions for collective action". In: *Governing the commons: the evolution of institutions for collective action* (1990). ISSN: 00237639. DOI: 10.2307/3146384.

[49] Elinor Ostrom. *Governing the Commons: The Evolution of Institutions for Collective Action*. Cambridge: Cambridge University Press, 1990.

[50] Elinor Ostrom. *Governing the Commons: The Evolution of Institutions for Collective Action*. Cambridge, UK: Cambridge University Press, 1990.

[51] Elinor Ostrom and T. K. Ahn. “he Meaning of Social Capital and its Link to Collective Action”. In: (2007). Available from: https://ssrn.com/abstract=1936058, [Accessed 12 December 2023].
[52] Elinor Ostrom and Toh-Kyeong Ahn. “The meaning of social capital and its link to collective action”. In: *Handbook of social capital: The troika of sociology, political science and economics* 17 (2009).
[53] PE Petruzzi, J Pitt, and D Busquets. “Electronic social capital for self-organising multi-agent systems”. In: *ACM Transactions on Autonomous and Adaptive Systems* 12 (2017), pp. 1–25. DOI: 10.1145/3124642. URL: http://dx.doi.org/10.1145/3124642.
[54] Jeremy Pitt. *Self-Organising Multi-Agent Systems*. WORLD SCIENTIFIC (EUROPE), 2021. DOI: 10.1142/q0307. eprint: https://www.worldscientific.com/doi/pdf/10.1142/q0307. URL: https://www.worldscientific.com/doi/abs/10.1142/q0307.
[55] Jeremy Pitt. *Self-Organising Multi-Agent Systems- Algorithmic Foundations of Cyber-Anarcho-Socialism*. 2021. URL: https://www.worldscientific.com/worldscibooks/10.1142/q0307#t=aboutBook.
[56] Jeremy Pitt. *Self-organising Multi-agent Systems: Algorithmic Foundations Of Cyber-anarcho-socialism*. 2021. DOI: 10.1142/q0307.
[57] Jeremy Pitt. *SOMAS L6: Electronic Instituions*. 2023. URL: ??.
[58] Jeremy Pitt. “Transforming Multi-Agent Systems into Self-Organising Electronic Institutions”. In: *Engineering Multi-Agent Systems*. Ed. by Huib Aldewereld and et al. Vol. 8758. Lecture Notes in Computer Science. Springer, Cham, 2014.
[59] Plato. *The Republic*. Available from: https://www.sciencetheearth.com/uploads/2/4/6/5/24658156/plato_-_the_republic.pdf, [Accessed 12 December 2023]. Translated.
[60] Pygame Developers. *Pygame Front Page — pygame v2.6.0 documentation*. https://www.pygame.org/docs/. Accessed: 2023-12-13. 2023.
[61] D. Ramirez-Cano and J. Pitt. “Follow the Leader: Profiling Agents in an Opinion Formation Model of Dynamic Confidence and Individual Mind-Sets”. In: *Intelligent Agent Technology (IAT), pp. 660–667* (2006). URL: https://ieeexplore.ieee.org/document/4052991.
[62] Daniel Ramirez-Cano and Jeremy Pitt. “Follow the Leader: Profiling Agents in an Opinion Formation Model of Dynamic Confidence and Individual Mind-Sets”. In: *2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*. 2006, pp. 660–667. DOI: 10.1109/IAT.2006.67.
[63] Anatol Rapoport. *Strategy and Conscience*. 2011.
[64] Nicholas Rescher. *Distributive Justice*. Indianapolis, IN: Bobbs-Merrill Company, Inc., 1966.
[65] Nicholas Rescher. *Distributive Justice: A Constructive Critique of the Utilitarian Theory of Distribution*. Available from: https://www.amazon.co.uk/Distributive-Justice-Constructive-Utilitarian-Distribution/dp/0829014152, [Accessed 12 December 2023]. 1966.
[66] Nicholas Rescher. *Distributive Justice: A Constructive Critique of the Utilitarian Theory of Distribution*. Bobbs-Merrill, 1967. DOI: 10.2307/2184188.
[67] Shalom H. Schwartz. “Universals in the Content and Structure of Values: Theoretical Advances and Empirical Tests in 20 Countries”. In: ed. by Mark P. Zanna. Vol. 25. Advances in Experimental Social Psychology. Academic Press, 1992, pp. 1–65. DOI: https://doi.org/10.1016/S0065-2601(08)60281-6. URL: https://www.sciencedirect.com/science/article/pii/S0065260108602816.
[68] John Scott. “Understanding Contemporary Society: Theories of The Present - Rational Choice theory-Complexity Theory”. In: *The British journal of sociology* 50.4 (2000). ISSN: 0007-1315.
[69] *SnakeViz Documentation*. https://jiffyclub.github.io/snakeviz/. Accessed: 2023-12-13.
[70] Jeff Suzuki. *Constitutional Calculus: The Math of Justice and the Myth of Common Sense*. 2015. URL: https://www.press.jhu.edu/books/title/11062/constitutional-calculus.
[71] Henri Tajfel and John C. Turner. *The Social Identity Theory of Intergroup Behavior*. 2004. URL: https://www.taylorfrancis.com/chapters/edit/10.4324/9780203505984-16/social-identity-theory-intergroup-behavior-henri-tajfel-john-turner?context=ubx.
[72] Robert L Trivers. “The evolution of reciprocal altruism”. In: *The Quarterly review of biology* 46.1 (1971), pp. 35–57.
[73] Robert L. Trivers. “The Evolution of Reciprocal Altruism”. In: *The Quarterly Review of Biology* 46.1 (1971). ISSN: 0033-5770. DOI: 10.1086/406755.
[74] Maximiliane Weiner. “Elinor Ostrom and Charlotte Hess (eds.). Understanding Knowledge as a Commons – From Theory to Practice”. In: *Journal of International and Global Studies* 2.2 (2011), p. 12. URL: https://digitalcommons.lindenwood.edu/jigs/vol2/iss2/12.
[75] Dr. Herbert I. Weisberg. *Willful Ignorance: The Mismeasure of Uncertainty*. 2014. URL: https://www.wiley.com/en-gb/Willful+Ignorance%3A+The+Mismeasure+of+Uncertainty-p-9780470890448.
[76] David H Wolpert and Kagan Tumer. “An introduction to collective intelligence”. In: *arXiv preprint cs/9908014* (1999).
[77] Everett L. Jr. Worthington. “The Role of Forgiveness in Rebuilding Social Capital”. In: *In: Saarni, C. and Harris, P.L. (Eds.), The Handbook of Emotional Development. Oxford University Press* (2001), pp. 508–527.