Universitat Oberta de Catalunya
Universitat Rovira i Virgili

MSc in Computational and Mathematical Engineering

Agent-Based Modeling of Human Behavior using PECS model and Game Theory

Author: Javier Navarro González
Tutor: Jordi Duch Gavaldà

Master's Thesis

Barcelona, July 15, 2019

Dr. Jordi Duch Gavaldà, certifies that the student Javier Navarro González has elaborated the work under his direction and he authorizes the presentation of this memory for its evaluation.

Director's signature:

Credits/Copyright

A page with the specification of credits / copyright for the project (either application on one hand and documentation on the other, or unified), as well as the use of brands, products or services of third parties (including source codes). If a person other than the author collaborated in the project, his identity must be made explicit and what he did.
The most usual case is exemplified below, although it can be modified by any other alternative:

Final Project Sheet

Title:	Agent-Based Modeling of Human Behavior using PECS model and Game Theory
Autor:	Javier Navarro González
Tutor:	Jordi Duch Gavaldà
Delivery date (mm/aaaa):	07/2019
Program:	Master in Computational and Mathematical Engineering
Area:	Computer Science and AI
Language:	English
Key words	Agent-Based Modelling, PECS, Predictability of Human Behaviour

Abstract

In this thesis we present a tool for studying human behavior using agent-based modeling combined with game theory.

With the help of the software *Netlogo*, and using the *PECS* reference model, we have implemented a tool for studying human behaviour based on decisions done playing to *Prisoner's Dilemma*, a classical game theory paradox, by using an agent-based model.

We have summarized the personality of agents in five different types, and by a given repertoire of possible actions, we have made agents play by using a set of rules for observing its behaviour. The experiment consist on see how agents decide whether cooperate or defect its opponent, based on their initial configuration and the influence of each component by assigning different weights, and to demonstrate the feasibility of a tool that combine game theory, PECS reference model and agent-based modelling for studying human behaviour and use it as starting point for future studies.

Resumen

En este proyecto presentamos una herramienta para estudiar el comportamiento humano usando un modelado basado en agentes combinado con teoría de juegos.
Con la ayuda del software *Netlogo* y usando el modelo de referencia *PECS*, hemos implementado una herramienta para estudiar el comportamiento humano basándose en decisiones hechas jugando al *Dilema del Prisionero*, una paradoja clásica de teoría de juegos, usando un modelado basado en agentes.
Hemos concentrado la personalidad de los agentes entre cinco tipos distintos, y dado un repertorio de acciones posibles, hemos hecho jugar a los agentes entre sí, usando un conjunto de reglas para observar su comportamiento.
El experimento consiste en ver como los agentes deciden entre cooperar o defectar a su oponente, basándose en su configuración inicial y en la influencia de cada coponente sobre ellos, asignándoles diferentes pesos, y para demostrar la factibilidad de una herramienta que combina teoria de juegos, el modelo de referencia PECS y modelado basado en agentes para estudiar el comportamiento humano y usarla como punto de partida en futuros estudios.

Resum

En aquest projecte presentem una eina per a la predicció del comportament humà fent ús d'un modelatge basat en agents combinat amb teoria de jocs.
Amb l'ajut del software *Netlogo* i utilitzant el model de referència *PECS*, hem implementat una eina per estudiar el comportament humà basant-nos en decisions presses jugant al *Dilema del Presoner*, una paradoxa clàssica de teoria de jocs, usant un modelatge basat en agents.
Hem concentrat la personalitat dels agents entre cinc tipus diferents, i donat un repertori d'accions possibles, hem fet jugar als agents entre sí, usant un conjunt de regles per a observar el seu comportament.
L'experiment consisteix a veure com els agents decideixen entre l'opció de confiar o defectar amb els seus oponents, basant-se en la seva configuració inicial i en la influència de cada component sobre ells, asignant-los diferents pesos, y per a demostrar la factibilitat d'una eina que combina teoria de jocs, el model de referència PECS i el modelatge basat en agents per a estudiar el comportament humà, i utilitzar-la com a punt de partida en estudis futurs.

Acknowledgements

I would like to thank my project director, Jordi for his guiding and supporting, for being patient even though sometimes it was not easy, and for helping me in any way possible during the realization of this project, providing me all the essential knowledge and material resources.

I would also like to thanks to my family, particularly to my parents, my sister and my friends for all the support, for encourage me and let me see that where there is a will there is always a way.

Contents

1. Introduction **1**
1.1 Motivation . . . 1
1.2 Context . . . 2
1.2.1 Areas of interest . . . 2
1.2.2 Stakeholders . . . 3
1.3 State of the art . . . 4
1.3.1 Reference models . . . 4
1.3.2 Previous studies . . . 7
1.4 Objectives . . . 8
1.5 Scope . . . 9
1.6 Project contribution . . . 9

2. Background **10**
2.1 Netlogo . . . 10
2.1.1 Basic concepts . . . 11
2.1.2 Sets of agents and functions . . . 13
2.2 Agent-Based Modelling . . . 18
2.2.1 Entities in Agent-Based modelling . . . 19
2.3 PECS model . . . 19
2.3.1 Structure of the agent world . . . 20
2.3.2 Environment . . . 20
2.3.3 Connector . . . 21
2.3.4 Structure of PECS agents . . . 21
2.4 Game Theory . . . 22
2.4.1 Basics . . . 22
2.4.2 Types . . . 23
2.4.3 Examples . . . 23
2.4.4 Limitations . . . 24

3. Initial architecture **25**
3.1 Overview . . . 25
3.2 Game theory . . . 25
3.2.1 Prisoner's Dilemma . . . 26
3.2.2 Payoff matrix . . . 26
3.2.3 Players phenotypes . . . 27
3.3 Agents . . . 30
3.3.1 Breeds . . . 30
3.3.2 Internal variables . . . 31
3.3.3 Setup . . . 31
3.4 System architecture . . . 32
3.4.1 Constants and variables . . . 33
3.4.2 Checks and setup . . . 34
3.4.3 Methods . . . 34
3.4.4 Data representation . . . 37

4. Final tool **38**
4.1 PECS integration . . . 38
4.1.1 Physic and Social Status . . . 38
4.1.2 Emotion . . . 39
4.1.3 Cognition . . . 40
4.2 Tool changes . . . 40
4.2.1 Variables . . . 40
4.2.2 Interface . . . 41
4.3 Final methods . . . 42
4.3.1 Move . . . 42
4.3.2 Possible actions . . . 42
4.3.3 Play . . . 45
4.3.4 Update variables and report data . . . 45

5. Experimentation and results **46**
5.1 Theoretical experiment . . . 46
5.2 Irrational experiment . . . 50
5.3 Rational experiment . . . 55
5.4 Balanced experiment . . . 59

6. Conclusions and Future Work **63**

Bibliography **64**

List of Figures

1 Illustration of human behaviour prediction 1
2 PECS model reference diagram 5
3 BDI model reference diagram 6
4 Netlogo desktop application 11
5 Netlogo possible shapes 16
6 Netlogo experiment setupscreen 18
7 Fire propagation example of ABM 19
8 PECS fundamental components 20
9 PECS structure and internal organization 21
10 PD Payoff matrix 26
11 Optimist phenotype heatmap 27
12 Pessimist phenotype heatmap 28
13 Envious phenotype heatmap 28
14 Trustful phenotype heatmap 29
15 Undefined phenotype heatmap 29
16 Slider variable number of individuals 31
17 Netlogo interface of our tool initial approach 32
18 Heatmap sample 37
19 Potential opponents based on neighbors 39
20 Final interface of our tool 41
21 Interface setup for first experiment 47
22 Heatmap of optimist breed 47
23 Heatmap of pessimist breed 48
24 Heatmap of envious breed 48
25 Heatmap of trustful breed 49
26 Heatmap of undefined breed 49
27 Heatmap of the aggregation of all breeds results 50
28 Interface setup for second experiment 51
29 Heatmap of optimist breed 51
30 Heatmap of pessimist breed 52

31 Heatmap of envious breed 52
32 Heatmap of trustful breed 53
33 Heatmap of undefined breed 53
34 Heatmap of the aggregation of all breeds results 54
35 Interface setup for third experiment 55
36 Heatmap of optimist breed 56
37 Heatmap of pessimist breed 56
38 Heatmap of envious breed 57
39 Heatmap of trustful breed 57
40 Heatmap of undefined breed 58
41 Heatmap of the aggregation of all breeds results 58
42 Interface setup for fourth experiment 59
43 Heatmap of optimist breed 60
44 Heatmap of pessimist breed 60
45 Heatmap of envious breed 61
46 Heatmap of trustful breed 61
47 Heatmap of undefined breed 62
48 Heatmap of the aggregation of all breeds results 62

List of Tables

1 Breed inherited commands 15

List of Abbreviations

ABM	Agent-Based Modeling
PECS	Physical, Emotional, Cognitive and Social Status
BDI	Belief-Desire-Intention
PD	Prisoner's Dilemma
C	Cooperate
D	Defect

1. Introduction

1.1 Motivation

Human behaviour is the response of individuals or group of humans to internal and external stimulation. It is influenced by biology and the environment in which someone is immersed, and the interaction of these two factors. But, it is possible to be predicted? Can we know why people do what they do, even before the person itself?

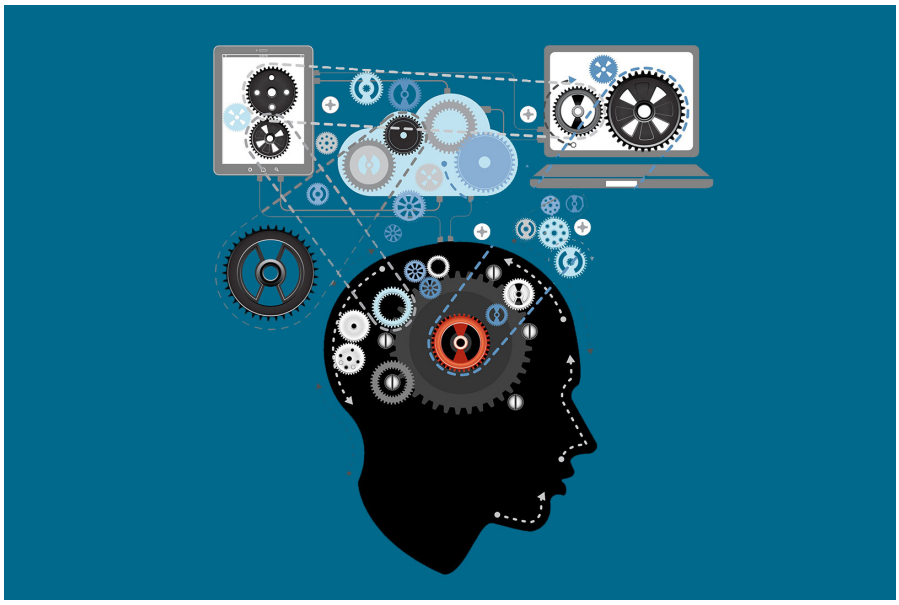

Figure 1: Illustration of human behaviour prediction

This question has been asked during years by hundred of scientific and psychologist, but still we do not have clear answers. Physics professor of *Boston University* Albert-László Barabási stands

that human behaviour is 93% predictable by studying mobility patterns of anonymous cell-phone users and concluded that [1], despite the common perception that our actions are random and unpredictable, human mobility follows surprisingly regular patterns. But, can it be extended to other situations?

The aim of this project is to develop a tool using agent-based modelling capable of predict human behaviour using the *PECS* reference model for the construction of human-like agents by enabling an integrative modelling of physical, emotional, cognitive and social influences within a component-oriented agent architecture and combined with game theory.

1.2 Context

This section is about the terms and concepts related to the project. We are going to make a description of the areas of interest as well as the stake-holders of the project: target audience, users and beneficiaries.

1.2.1 Areas of interest

We can distinguish three main areas of interest in the project. The first area is human behaviour and its predictability, which is related to the psychology field, the second one is about agent-based modeling, a technique for running simulations with a set of features and the third one is game theory field.

Predictability of human behaviour

In this project we are trying to develop a tool capable of predict human behaviour, which is the capacity of mental, physical, emotional and social activities experienced during the five stages of a human being's life - prenatal, infancy, childhood, adolescence and adulthood.

For sociologists, most of this behaviour is predictable. Spontaneous individuals are largely absent from the population and human mobility follows surprisingly regular patterns, free will is not absolute. Suggestion plays an important role on human behaviour, because we cannot completely control our thoughts, what makes behaviour predictable [3].

For answer this question, we are going to use the most well-know model for representing human behaviour, and setting up its input and output functions and weighting the different components, we are going to see how affects to agents behaviour. By using common features shared by all person and a limited set of actions, we are going to simulate what agents will do depending on its capacities and develop a tool for observe if they do what we think that agents will do.

Agent-Based modeling

For developing the project, we need to generate a model for doing simulations and analyze its results. Agent-based modeling (ABM) [4] focuses on the individual active components of a system.

With ABM, active entities, known as **agents**, must be identified and their behaviour defined. They may be people, house holds, vehicles, products, etc. whatever is relevant to the system. Connections between them are established, environmental variables set, and simulations run. The global dynamics of the system then emerge from the interactions of the many individual behaviours.

ABM helps in understanding how simple micro-rules of individual behaviour emerge into macro-level behaviour of a system. Being able to model these complex systems can lead to a better understanding of them, thereby being able to control the course of events, just by tweaking simple rules at the individual level.

Games Theory

Game theory is the study of mathematical models of strategic interaction between rational decision-makers, developed during 50s, that has been applied into different fields as social science, logic, computer science or biology [5].

It has a very general scope, encompassing questions that are basic to all of the social science. It can offer insights into any economic, political or social situation that involves individuals who have different goals or preferences.

From all possible game types that offers game theory, we are going to focus in the *Cooperative / Non-cooperative* games. A game is cooperative if the players, agents in our case, are able to from binding commitments externally enforced, meanwhile a game is non-cooperative if players cannot form alliances or if all agreements need to be self-enforcing.

This kind of games focuses on predicting which coalitions will form, the joint actions that groups take and the resulting collective payoffs. It is opposed to the traditional no-cooperative game theory which focuses on predicting individual players' actions and payoffs and analyzing *Nash equilibria*.

1.2.2 Stakeholders

The development of this project can draw the attention of some people from different areas. These are the potential stakeholders and the reasons why they can be interested in our project.

Target audience

Our target audience can be either in business side or in an non-profitable way such in the academical world.

In the first case, by predicting human behaviour, companies can anticipate a person needs even before itself. Thus, it makes a competitive advantage for developing non-existing products by learning from its future clients needs or by improving an existing product.

On the other hand, learn from people how they 'work' can be the key for understanding spread of epidemics, population dynamics, vegetation ecology, landscape diversity, the growth and decline of ancient civilizations, mental diseases, or even eviscerate key concepts such *love*.

Users

The aim of the project is to implement a tool that can be able to predict human behaviour and understand better how human beings make its decisions. Our project can be extrapolated to other areas, which generate a lot of potential users. This potential users will be in fact any company, because human behaviour condition any decision of a human being.

Beneficiaries

Besides the final users, this project can be beneficial for investigators on the topic. They can take benefit of the investigation done during the development of this project and improve the results or add new features using our tool as starting point.

1.3 State of the art

In this section, we are going to give information about the state of the art of the different elements related to our project. We are going to describe the most important reference models and the latest techniques used in our interest fields. Moreover, we are going to summarize previous studies related with our project and discuss about them.

1.3.1 Reference models

As we have said above, there are several reference models. Coming up next, we are going to cite the two closest to our problem.

Physical, Emotional, Cognitive and Social Status (PECS)

PECS [6] is a reference model which makes it possible to specify and to model physical, emotional, cognitive and social factors and their interactions. Is a multi-purpose reference model for the simulation of human behaviour in a social environment.

PECS agent architecture grounds on the opinion that an agent must be capable of integrating physical, emotional, cognitive and social attributes and processes in order to provide an adequate means for modelling human behaviour.

The architecture may be divided up into three different horizontal layers. The input layer consists of the componets *Sensor* and *Perception* and is responsible for the processing of input data coming from the environment of the agent.

The internal layer is composed by the components *Physics, Emotion, Cognition* and *Social Status*. This group of components models the internal state of the agent.

Finally, in the output layer, covering the components *Behaviour* and *Actor*, the behaviour of the agent is calculated and the actions are executed.

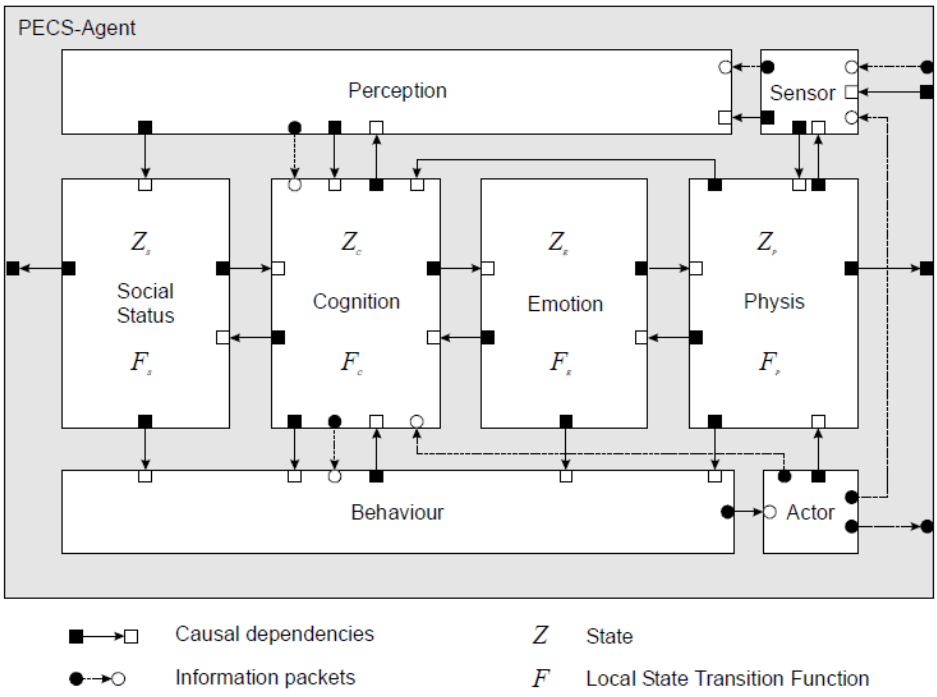

Figure 2: PECS model reference diagram

Belief–Desire–Intention (BDI)

BDI [7] is a software model developed for programming intelligent agents. BDI stands for *Belief, Desire* and *Intention*, and the original principles were set by *Michael Bratman* during the 80s. In essence, it provides a mechanism for separating the activity of selecting a plan from the execution of currently active plans.

BDI as whole can be represented by the following components:

- **Belief**: The knowledge of the world, *state of the world*.
- **Desire**: The objective to accomplish, *desired end state*.
- **Intention**: The course of actions to achieve the desire of the agent.

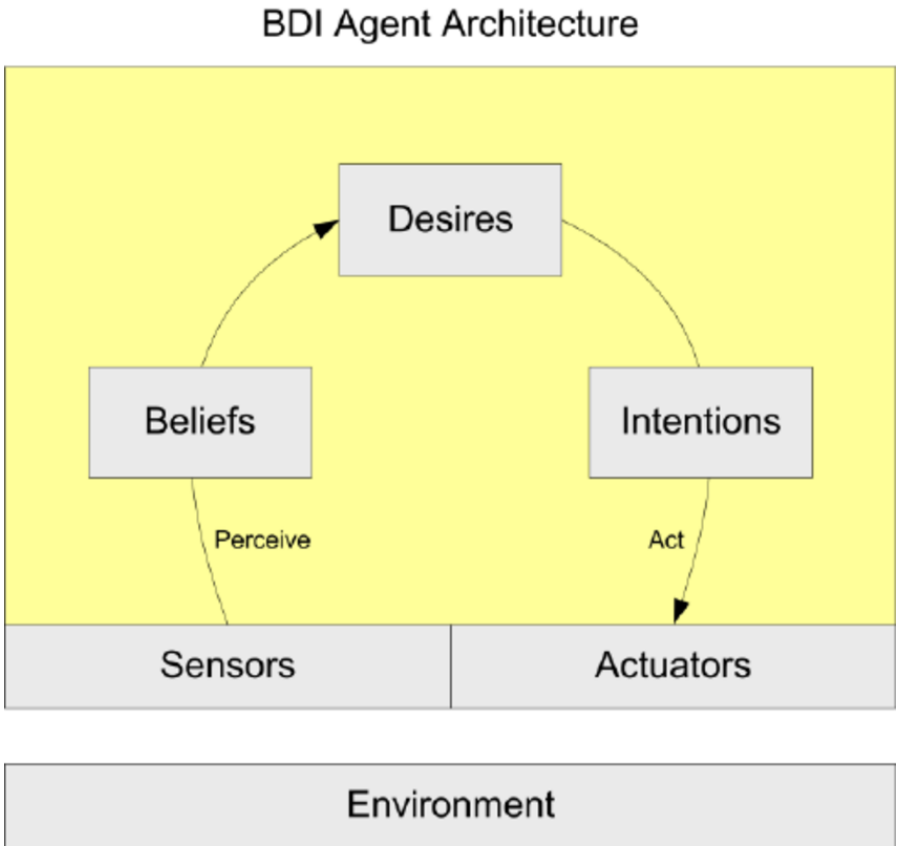

Figure 3: BDI model reference diagram

The BDI software model is closely associated with intelligent agents, but does not, of itself, ensure all the characteristics associated with such agents. For example, it allows agents to have private beliefs, but *does not force them to be private*. It also has nothing to say about agent communication. Ultimately, the BDI software model is an attempt to solve a problem that has more to do with plans and planning (the choice and execution thereof) than it has to do with the programming of intelligent agents.

1.3.2 Previous studies

Starting our project, we had taken into account previous works and studies related. In the next sections, we are going to summarize the three most relevant studies, the ones that are close to our project, but there are also another important works, as *Predicting Human Behaviour in Normal-Form Games* [8], that give to us very useful information.

PECS: Agent-Based Modelling of Human Behaviour

This work [9] from Christoph Urban and Bernd Schmidt done at University of Passau, is a base point for understanding the used reference model *PECS*.

In this project it is presented the PECS reference model for the construction of human-like agents, which enables an integrative modeling of physical, emotional, cognitive and social influences within a component-oriented agent architecture.

Furthermore, the case study *Learning Group* is introduced which demonstrates the practical application of the design methodology, and is intended to model basic social psychological mechanisms in the context of group formation and disbandment.

Modelling of Human Behaviour. The PECS Reference Model

In this second work [10] from the University of Passau and again done by Dr. Bernd Schmidt, PECS model is detailed. Different components of the model are explained deeply for a better understand of how it model an agent behaviour.

PECS is presented as a reference model for the modelling of human behaviour. The architecture proposed claims to be universally applicable, and adaptation to individual conditions occurs by means of filling in the empty spaces provided by the architecture.

Agents can be endowed with a varied repertoire of actions that state the external actions of which the agent is capable. As a result, very diverse agents and agent communities develop but they all have the same deep structure and therefore they can all be described by one and the same reference model.

Humans display a reduced set of consistent behavioral phenotypes in dyadic games

The article [11] questions about people behavior when play games, precisely when are exposed to the constraint introduced in game theory designs, where people are often not "rational" in the sense that they do not pursue exclusively self-interested objectives .

This behavior is especially clear in the case of prisoner's dilemma, where rational choice theory predicts that players will always defect but empirical observation shows that cooperation oftentimes occurs, even in "one shot" games where there is no expectation of future interaction among the parties involved.

These findings beg the question as to why players sometimes choose to cooperate despite incentives not to do so, questioning if these choices are a function of a person's identity and therefore consistent across different strategic settings, if individuals draw from a small repertoire of response, and if so, what are the conditions that lead them to choose one strategy over another.

For that reasons, an empirical experiment has been done over more than 500 people, with prizes in form of incentives for assuring that people with try to do whatever to win if they are motivated to. There are 5 phenotypes of person and it is studied differences between each phenotype from theoretical expected results to empirical collected results.

1.4 Objectives

The main goal of the project is simulate human behaviour by building a tool based on agents modelling, applying game theory and with PECS reference model embedded for handling its behavior. For doing that, we are going to implement a model where agents plays a classic games theory paradox, the *prisoner's dilemma*.

We are going to bring agents with a repertoire of actions and rules, which are going to influence its behaviour, even though sometimes they are not going to pick the best option, from a rational point-of-view, due to the impact of emotions and cognition, which are going to be represented in form of revenge or the best option based on previous games that the agent have played.

Hence, objectives can be divided in:

- **Different phenotypes**: Each agent as the representative of a human being will be endowed with an elementary personality that will influence its behaviour. We need to have a number of different personalities very accurate to no interfere on the experiment results.
- **Set of actions**: Alongside the different personalities, we need to endow to the agent with a repertoire of possible actions of an agent. The number of actions must be kept to the strict minimum for simplicity reasons, but the necessary ones to have realistic results.

- **Agent behaviour**: By using different personalities and a limited set of actions, we have to manifest the agent behavior. This is a key point because if we do not define well all the possible situations and we leave some corner cases, we cannot obtain representative results.

1.5 Scope

In order to achieve the proposed objectives, we need to implement a program capable of determine different agents depending on its personalities, able them with a set of actions and manifest it with a coherent behaviour on each decision made.

We are going to implement this model in *Netlogo* [13], which will carry the simulations of the agent over the time and develop the experiment. The program brings a visual tool for watching on real time the simulation and also the possibility of represent the results in plots or by 'asking' to the agents.

Also, we are going to use the previous mentioned works *PECS – Agent Based Modelling of Human Behaviour* and *Humans display a reduced set of consistent behavioral phenotypes in dyadic games* as starting point. Both will bring us needed information about agents phenotypes and reference model to define its behavior.

The final result will be a set of agents with different personalities wandering in a defined universe, interacting with other agents to achieve its internal needs and goals, and we are going to be the observers that take notes and study how they interact for extracting conclusions about human behaviour without interfere in the result of the experiment.

1.6 Project contribution

The contribution of this project is to develop and show the feasibility of a power tool that combines game theory, PECS reference model and agent-based modelling that can be scaled and tuned to predict human behaviour in a particular case, to understand better why human beings do what they do or how they make its decisions. The results reached from our tool can be easily extrapolated to other fields, so in fact, we want to contribute on something that is global, as can be understand better how human brain works.

Since our tool has been done with academical purposes, all code is open source, as well as all frameworks, packages and other software used in its development.

2. Background

Before starting to explain the project development, it is necessary to have some background on the related topics of the project. Therefore, in this chapter we provide a definition and the essential knowledge of the techniques and models used during the development of the project.

2.1 Netlogo

Netlogo is a programming language adapted to the modelling/simulation of phenomenons where appears a large number of individuals interacting between each others. It is called Netlogo because it is a dialect of the *Logo* language [14], and is a multi-agent programmable modelling environment that:

- Is adequate for modelling complex systems which evolve during the time.
- Is adequate for modelling hundred or thousands of individuals (persons, bacteria, insects, organizations, graph nodes, etc.
- Allow explore connections between local interactions at individual level and the macroscopic patterns that emerge from them.

It is also an easy and intuitive programming environment for create and testing new models:

- Allows open and experiment simulations.
- Allows implement models fast for checking hypothesis about decentralized systems.
- Granted with a large library of simulations in natural and social sciences, which ones can be used and modified.
- Models are built using a simple language inspired in functional/declarative paradigm.
- Have an intuitive graphical interface.

It has a Desktop application with the complete feature set of functionality but also has a web version for running simulations in browsers, even though it is not available full functionality.

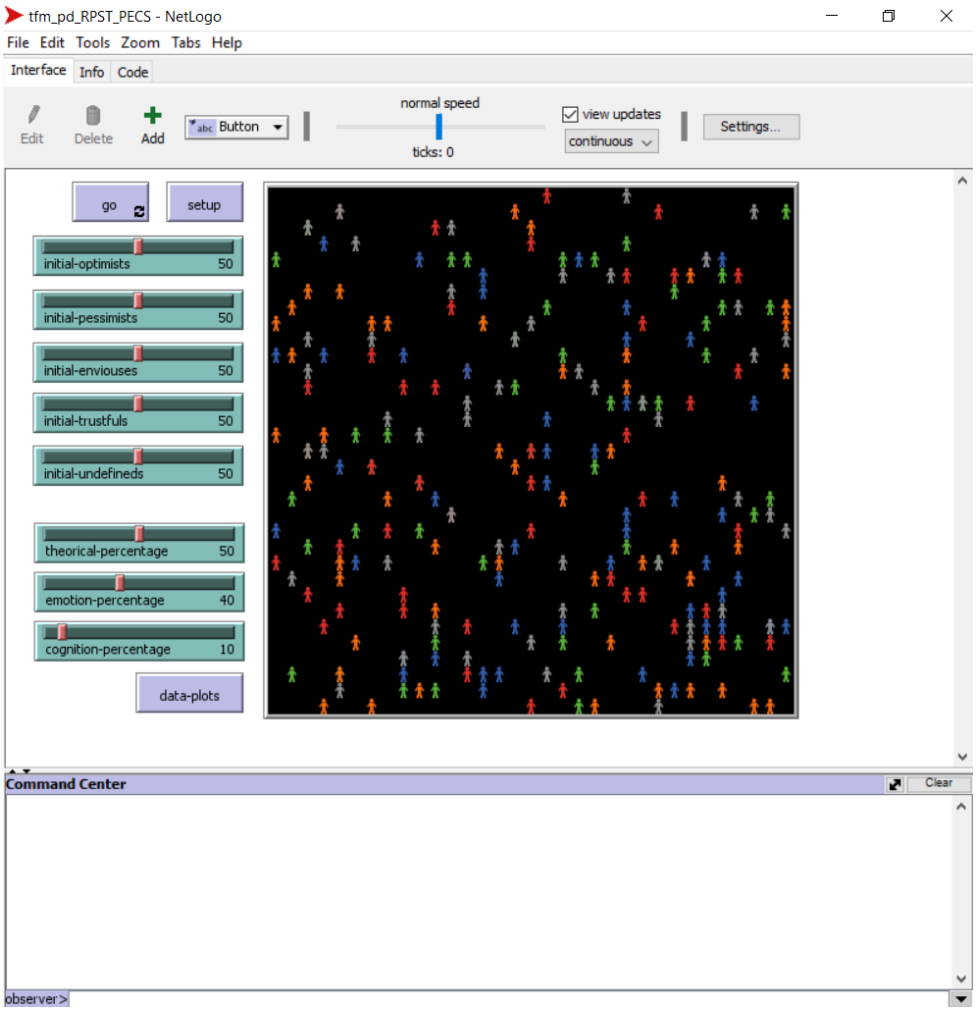

Figure 4: Netlogo desktop application

2.1.1 Basic concepts

Even though Netlogo language is easy to learn, it does not set limits to the sophistication of the models that can be implemented. It has as main orientation modelling systems composite by individuals that interact between each others and with the environment, based on the **agent modelling paradigm**:

- An **agent** is an synthetic individual, autonomous and doted with rules or features that govern its behaviour and its capacity for making decisions.
- Agents interact between each others and with the environment obeying a set of rules.
- Agents are flexible and have the ability of learning and adapt its behaviour based on the experience. This ability requires some way of memory. Agents can even have rules for modifying its behaviour rules.

Agents

Netlogo is equiped with for types of agent:

- **Mobile agents (turtles)**: They are the ones that are moving 'around the world'. The world is in 2D (even though there is a 3D version) and it is divided in a mesh of patches.
- **Immobile agents (patches)**: Every square division of the world.
- **Linking agents (links)**: Agents that connect between each others the turtles (like edges in a graph)
- **Observer agent**: It is not located and it can interact with all the elements of the world. In a way, it represents the super agent that can control all the elements of the world.

We can control a Netlogo model by using buttons and switches. Furthermore, the system offers the possibility of controlling the models by a command center, where it is possible execute commands about a model and modify its parameters.

Variables

In Netlogo we can either user defined framework variables or create our own. This own-created variables can be global when represents a global system property (i.e. number of generations formed), variables that belongs to *patches* and are used for representing memory attributes for each portion of 'universe' (i.e. altitude, amount of food allocated in that slice of 'universe'), or variables that belongs to the *turtles* and represent equally memory attributes for each one of the turtles (i.e., type, gender or last occupied position).

We are going to show some examples of these three type of variables that can be found in *Netlogo*:

- **Patch variables**: *patches-own [altitude town]*
- **Turtle variables**: *turtles-own [gender last-position]*

- **Global variables**: *globals [generation]*

These commands only works on *Procedures* tab, where it is implemented the model code.

2.1.2 Sets of agents and functions

Sets of agents

In this section we are going to use turtle's gender for modifying its visual aspect. One way of doing this consist on divide turtles by its gender variable. For that, we can use the command:

```
ask turtles with [ gender = 0 ]
    [ set color pink
]
```

What previous command achieves is to execute a determined action over the subset of turtles which verifies a certain property.

Netlogo bring very powerful tools for applying commands selectively over a set of agents that verify a certain property by using **ask set-of-agents [commands]**. This is the general way of the *ask* instruction, but previously we have applied to the particular case in which we consider as set of agents from the total of turtles or patches.

Some examples of how implement set of agents are:

- ***turtles with [gender = 1]***: *Turtles* with *gender* 1.
- ***turtles with [xcor > 0]***: *Turtles* on the right side of the 'universe'.
- ***patches with [pycor > 0]***: *Patches* on the north 'hemisphere'.
- ***turtles-here***: All the *turtles* in a certain patch. This form must be used by another *turtle* or *patch*.
- ***other turtles***: Similar to *turtles*, but in this case the *turtle* that executes the call is not included.

Control structures

We are going to change the visual appearance of turtles with *gender* 1. On this case, we are going to use control structures for illustrate its use. For that, we can execute the command

```
ask turtles [
    if (gender = 1) [ set color pink ]
]
```

Control structures offer another way for executing selectively actions in *Netlogo*:

- **if condition** [**commands**]: If the condition is verified, *commands* are executed:

```
if (gender = 1) [ set color sky ]
```

- **ifelse condition** [**commands1**] [**commands2**]: If the condition is verified, *commands1* are executed, if not *commands2*:

```
ifelse
    (gender = 0) [ set color pink ]
    [ set color sky ]
```

- **ifelse condition1** [**commands1**] **condition2** [**commands2**] **...**: If *condition*-i is verified, *commands*-i are executed, if not, else-*commands*:

```
ifelse
    (gender = 0) [ set color pink ]
    (gender = 1) [ set color sky ]
    [ set color green ]
```

- **while** [**condition**] [**commands**]: Meanwhile *condition* is verified, commands are executed:

```
ask turtles [
    while [ycor < 0 ]
        set ycor (ycor + 1) ]
    ]
]
```

Breeds

In the system we are not limited to the use of one and only specie of mobile agents (*turtles*), we can create many as we want, each one with its own attributes and behaviours. On that way, we

can imitate ecosystems composed by some species or sophisticated economies formed by different types of producers and consumers.

For creating and specie, it is used the command

```
breed [ plural-specie singular-specie ]
```

On this way, we can refer to the whole specie or to one of its individuals. Some commands and reports will use the plural name of the breed, for instance *create-*, meanwhile others will use singular version, *-neighbor* (environment of the individual). In general, the rules that follows for using one or another are the derived ones of the natural language.

Once defined a new breed, automatically it is generated a set of agents associated to itself, in the way that everything that we have seen for sets of agents in general can be applicable for selecting individuals of the same breed.

For instance, for creating two breeds (fishes and frogs) we will use commands:

```
breed [ fishes fish ]
breed [ frogs frog ]
```

Once created, all new breeds have equivalent commands to the *turtles* ones, that are inherited automatically. For using them it is enough with substitute, on each command, the word *turtles* by the name of the breed:

Turtles commands	**Breed commands**
create-**turtles** n	create-**fishes** n
ask **turtles** [commands]	ask **fishes** [commands]
ask **turtles** hatch n [commands]	ask **fishes** hatch n [commands]
count **turtles**	count **fishes**
ask **turtles** with [condition] [commands]	ask **fishes** with [condition] [commands]
turtles-here	**fishes**-here
other **turtles**	other **fishes**
turtles-own	**fishes**-own

Table 1: Breed inherited commands

Moreover, each breed inherit the following primitives automatically: **create-**, **hatch-**, **sprout-**, **-here**, **-at**, **-on** and **is-?**.

Even though, individuals of a breed can migrate to another breed. For instance a fish do not must be fish the rest of its life, and it could transforms to an amphibian (supposing that we have created both breeds), by properly modifying its *breed* property that all agents have:

```
ask one-of fishes [ set breed frogs ]
```

As visual enforcement, *Netlogo* offers also a way of modifying the aspect of diverse turtles families by using command:

```
ask turtles [ set shape name-of-shape ]
```

Figure 5: Netlogo possible shapes

Actions and functions

Netlogo allows the construction of our own commands. The most common use is **actions**, which are like scripts that group in logic units a set of commands that is repeated commonly, which makes it easier its execution and modification in case of necessity. The are defined in the following way:

```
to name-action [ parameter1 parameter2 ... parametern ]
    command-1
    command-2
    ...
    command-n
end
```

For calling a routine, it is used straight its name with its associated input parameters. The **methods** are another way of grouping commands, with the difference that a method **returns a value**.

```
to-report name-method [ parameter1 parameter2 ... parametern ]
    command-1
    command-2
    ...
    command-n
    report val
end
```

Experiments

BehaviorSpace is one of the integred tools in *Netlogo* that allows make experiments about the models. Its basic function is to run the model several times, modifying systematically the parameters of the model and recording the results of each execution.

It is common that models have a lot of parameters which depends on, each one with its range. It is recurrent that the modification of this mentioned parameters provoke a drastic change on the behaviour of the system which is being modeled. In complex systems it is specially hard to know before an execution the values of the parameters that the model is going to follow.

As simple example we can take the fire propagation through a forest model. For instance, the fire starts from the left side of the 'universe' and it propagates by contacting with trees. In the model we just have one parameter which is tree density in the forest (between 0 and 100) and we check the evolution through the proportion of burnt trees. We can observe that the model presents a very abrupt transition phase, in which, for certain values of density (above a very thigh threshold), the fire propagation is very low, meanwhile with values above a certain threshold, the final proportion of burnt trees is around 100%.

Of course, we can do several executions of the model modifying manually density, but this is a tedious process when the space of parameters is too big. Is here where play an important role this tool, which allows us to automatize this process and storing the interesting results in a file to be analyzed later with the proper tool.

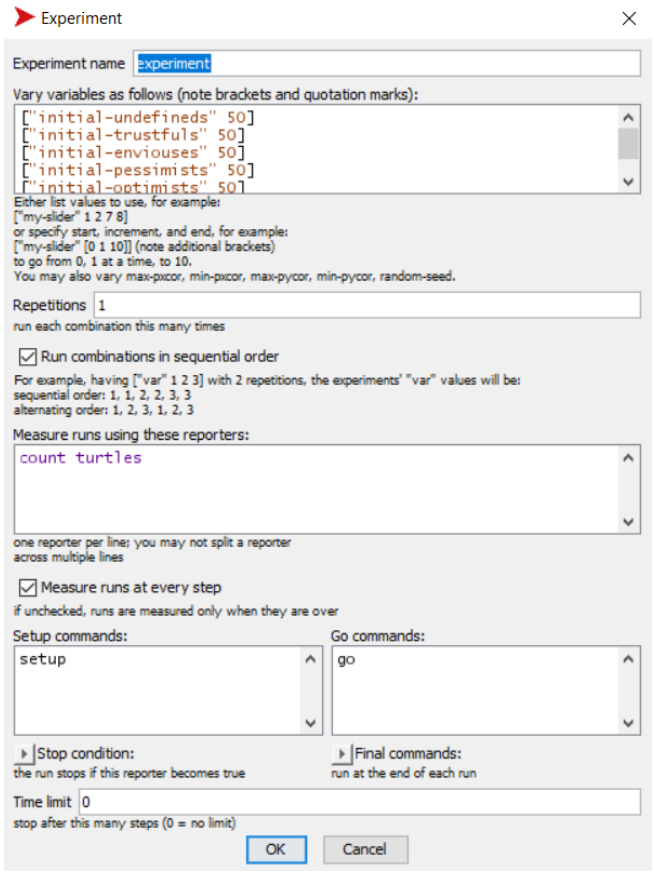

Figure 6: Netlogo experiment setupscreen

2.2 Agent-Based Modelling

Agent-Based Modelling (ABM) [15] is a style of modelling in which we represent the interaction between individuals and with each other environment in a program. Agents can be, for example, people, animals, groups, or cells. They can model entities that do not have a physical basis but are entities that can perform some tasks such as gathering information or modelling the evolution.

ABM is a method of modelling complex systems by defining rules and behaviours for individual components (agents) as well as the environment thy are present in. Further, we aggregate these rules to see the general behaviour of the system. It helps in understanding how simple micro-rules of individual behaviour emerge into macro-level behaviour of a system. Being able to model these complex systems can lead to a better understanding of them, thereby being able to control the course of events, just by tweaking simple rules at the individual level.

2.2.1 Entities in Agent-Based modelling

ABM contains autonomous models called *agents*. These agents can be an individual, a group of individuals, or even an organization. Each agent is defined with properties of its own along with relationships with other agents.

Apart from agents, ABM also have environments, which is a set of conditions the agent is exposed to. Once these entities are defined in an ABM, individual behavioral rules of how an agent would behave in a given environment is defined.

An aggregate of these simple individual-level behaviours that lead to a complex macro level patter. A feedback-based-learning-model is often used in ABMs to update agent actions based on their changing relationships with other agents and their environment.

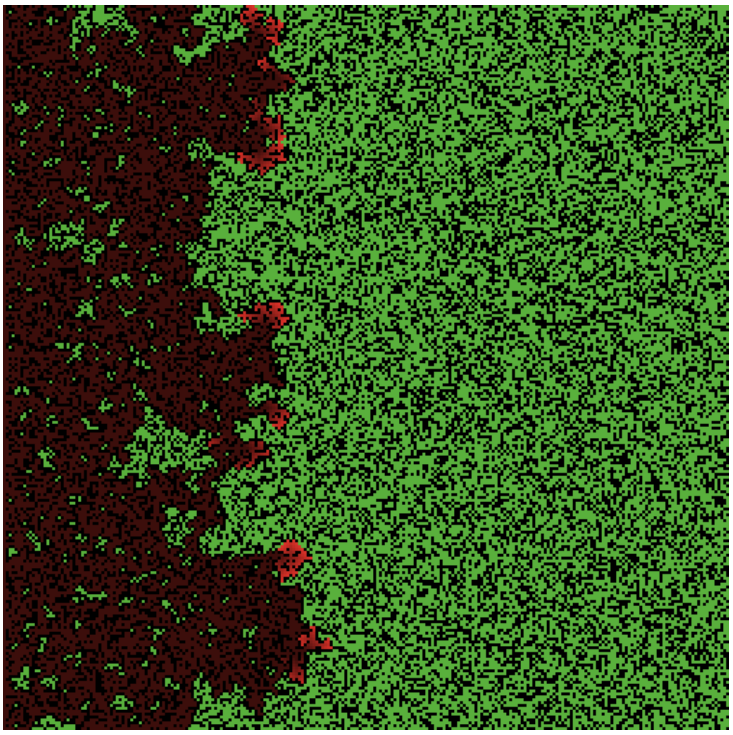

Figure 7: Fire propagation example of ABM

2.3 PECS model

PECS is a reference model which makes it possible to specify and to model these factors and their interactions. A detailed description of the PECS reference model and its underlying methodology.

PECS stands for: **P**hysical conditions, **E**motional state, **C**ognitive capabilities and **S**ocial status.

We are going to describe into five subsections the PECS reference model.

2.3.1 Structure of the agent world

The agent world of the reference model PECS consists of the following fundamental components:

- The environment component
- The connector component
- The agents

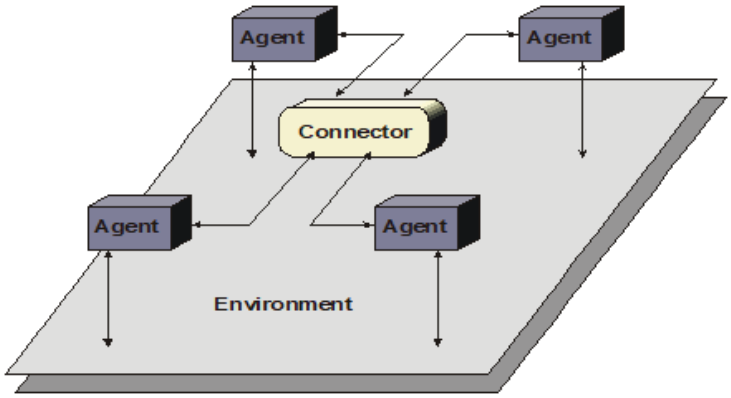

Figure 8: PECS fundamental components

2.3.2 Environment

The environment component is used to model external events and influences which are important for the behaviour and the actions of agents. It may also include other agents.

It is also reflected in the agents' knowledge. The environmental knowledge that an agent has and that is represented in the model it has made of the environment may be incomplete, uncertain and even erroneous. As the environment is generally very diverse and difficult to squeeze into a uniform schema, PECS reference model provides little assistance in this respect. It merely provides a frame within which the agents can move. The individual design of the environment component is a mater for the user alone.

2.3.3 Connector

Communication is an essential aspect of multi-agent systems. The basic principle is that all agents must have the possibility of communicating with all other agents. For this purpose the *Connector* component is introduced, which serves as a central switchboard that organizes the exchange of information between agents. Again, its individual design is a matter for the user

2.3.4 Structure of PECS agents

The basic structure of PECS model is clearly recognizable: Based on system theory, consists of input, internal state and output.

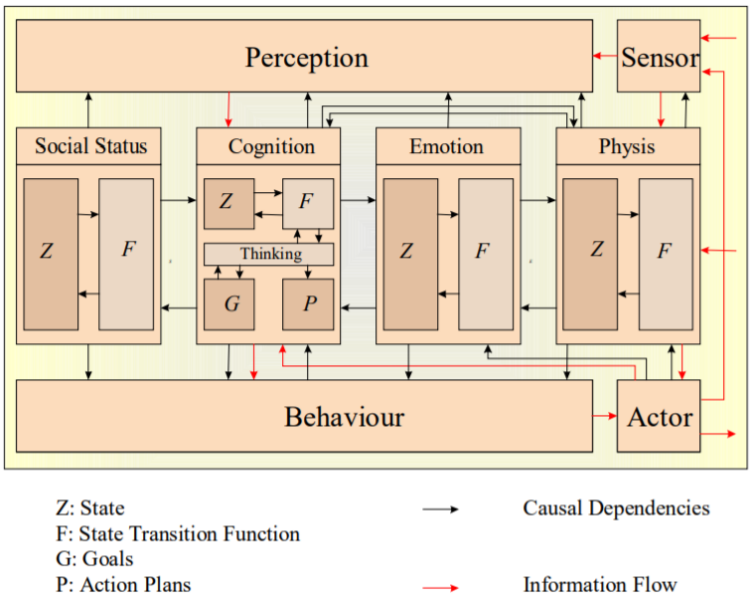

Figure 9: PECS structure and internal organization

The upper level with the components *Sensors* and *Perception* corresponds to the input. These components are responsible for the reception and initial processing of information from the environment.

The middle four components *Status*, *Cognition*, *Emotion* and *Physis* contain the state variables of the agent and their changes of state.

The components at the bottom of the diagram *Behaviour* and *Actor*, are responsible for output. The *Behaviour* component determines the execution order. It contains the set of rules on the basis of which an execution order is issued as it were automatically.

The execution orders are passed to the *Actor*, who is responsible for their execution. The *Actor* contains the full repertoire of actions of which the agent is capable.

2.4 Game Theory

Game theory [16] is a theoretical framework for conceiving social situations among competing players. In some respects, game theory is the science of strategy, or at least the optimal decision-making of independent and competing actors in a strategic setting. The key pioneers of game theory were mathematicians John von Neumann and John Nash, as well as economist Oskar Morgenstern.

2.4.1 Basics

The focus of game theory is the game, which serves as a model of an interactive situation among rational players. The key to game theory is that one player's payoff is contingent on the strategy implemented by the other player. The game identifies the players' identities, preferences, and available strategies and how these strategies affect the outcome. Depending on the model, various other requirements or assumptions may be necessary.

Game theory has a wide range of applications, including psychology, evolutionary biology, war, politics, economics, and business. Despite its many advances, game theory is still a young and developing science.

The Nash Equilibrium

Nash Equilibrium [17] is an outcome reached that, once achieved, means no player can increase payoff by changing decisions unilaterally. It can also be thought of as "no regrets," in the sense that once a decision is made, the player will have no regrets concerning decisions considering the consequences.

The Nash Equilibrium is reached over time, in most cases. However, once the Nash Equilibrium is reached, it will not be deviated from. After we learn how to find the Nash Equilibrium, take a look at how a unilateral move would affect the situation. Does it make any sense? It shouldn't, and that's why the Nash Equilibrium is described as "no regrets." Generally, there can be more than one equilibrium in a game.

However, this usually occurs in games with more complex elements than two choices by two players. In simultaneous games that are repeated over time, one of these multiple equilibrium is reached after some trial and error. This scenario of different choices over time before reaching

equilibrium is the most often played out in the business world when two firms are determining prices for highly interchangeable products, such as airfare or soft drinks.

2.4.2 Types

Although there are many types (i.e. symmetric/asymmetric, simultaneous/sequential, et al.) of game theories, cooperative and non-cooperative game theories are the most common. Cooperative game theory deals with how coalitions, or cooperative groups, interact when only the payoffs are known. It is a game between coalitions of players rather than between individuals, and it questions how groups form and how they allocate the payoff among players.

Non-cooperative game theory deals with how rational economic agents deal with each other to achieve their own goals. The most common non-cooperative game is the strategic game, in which only the available strategies and the outcomes that result from a combination of choices are listed. A simplistic example of a real-world non-cooperative game is Rock-Paper-Scissors.

2.4.3 Examples

There are several "games" that game theory analyzes. Below, we will just list a few of these, only describing the one that we use in our thesis, the *Prisoners' Dilemma*

- **Dictator Game**
- **Volunteer's Dilemma**
- **Centipede Game**
- **Prisoner's Dilemma**

The Prisoner's Dilemma is the most well-known example of game theory. Consider the example of two criminals arrested for a crime. Prosecutors have no hard evidence to convict them. However, to gain a confession, officials remove the prisoners from their solitary cells and question each one in separate chambers. Neither prisoner has the means to communicate with each other. Officials present four deals, often displayed as a 2 x 2 matrix.

1. If both confess, they will each receive a five-year prison sentence.
2. If Prisoner1 confesses, but Prisoner2 does not, Prisoner1 will get three years and Prisoner2 will get nine years.

3. If Prisoner2 confesses, but Prisoner1 does not, Prisoner1 will get 10 years, and Prisoner2 will get two years.

4. If neither confesses, each will serve two years in prison.

The most favorable strategy is to not confess. However, neither is aware of the others strategy and without certainty that one will not confess, both will likely confess and receive a five-year prison sentence. The Nash equilibrium suggests that in a prisoner's dilemma, both players will make the move that is best for them individually but worse for them collectively.

2.4.4 Limitations

The biggest issue with game theory is that, like most other economic models, it relies on the assumption that people are rational actors that are self-interested and utility-maximizing. Of course, we are social beings who do cooperate and do care about the welfare of others, often at our own expense. Game theory cannot account for the fact that in some situations we may fall into a Nash equilibrium, and other times not, depending on the social context and who the players are.

3. Initial architecture

3.1 Overview

The goal of this initial point is to build the main system that will place our simulations. On this incipient part of the project development, we want to setup a scalable structure for creating agents with different personalities, with a set of variables that will rule their fate and a certain number of actions to develop.

Also, we want to introduce game theory as the key concept to evaluate human behaviour. We are going to make our agents play between each others randomly to the *Prisoner's Dilemma*, a standard example of a game analyzed in game theory that shows why two completely rational individuals might not cooperate, even if it appears that it is in their best interests to do so.

We need to build an ecosystem to handle our game and agents too. The system is going to be built using *Netlogo* framework, which is very accurate for agent-based modelling and brings us all needed tools for handling our simulations. The system will be the universe that contains our agents, and will act as an observer, giving us visual and data feedback about agents chances.

3.2 Game theory

As we have said before, game theory will be the way that we use to evaluate agents decisions. We are going to make agents play against each other to *Prisoner's Dilemma* (PD), which is a game theory classical paradox. We use PD as a way for evaluate agent decisions and so, include a reward in form of a payoff matrix to make agents try to maximize the number of points that they can achieve based on its decisions.

All the data used in the project about the phenotypes, payoff matrix (scoring system) and the dynamics of PD is based on article *Humans display a reduced set of consistent behavioral phenotypes in dyadic games* [11].

3.2.1 Prisoner's Dilemma

In our project, we have implemented classical PD game version, but changing the payoff matrix values. Agents will play against each other and they will have to choose between **Cooperate (C)** or **Defect (D)**, and depending on its decision and rival's decision, they will obtain a score.

The decision is conditioned by agent personality, because it will impact in the way how to make it. By combining different agents personalities and a changing payoff matrix every round, we are creating a situation to check every agent strategy.

3.2.2 Payoff matrix

The payoff matrix will be crucial for making agents to change its behaviour depending on the values that it holds. As we have pointed above, some values of the matrix are **variable**, meanwhile than others are constants. This is the payoff matrix used in our tool:

$$\begin{array}{cc} & \begin{array}{cc} C & D \end{array} \\ \begin{array}{c} C \\ D \end{array} & \begin{pmatrix} R & S \\ T & P \end{pmatrix} \end{array}$$

Figure 10: PD Payoff matrix

This way, we have all possible combinations generated from the choices of two agents (four possible combinations). The payoff matrix form is interpreted as the rows are agent's strategies, whereas columns are those of its opponent. In that way, an agent will have one of the four possible values as score every round, stated by variables ***R, S, T*** and ***P***.

As we have mentioned before, not all values of the payoff matrix are constant. The ones on the **main diagonal** will be **constant**, while the ones on the **secondary diagonal** will be **variable**. This way, will have **constant** values for ***R*** and ***P***, while values for ***S*** and ***T*** are **variable**. These are their assigned values:

- **R**: Result of both agents C has a fixed value of **10**.
- **P**: Result of both agents D has a fixed value of **5**.
- **S**: Result of first agent C and second agent D has a variable value in range **[0,10]**.
- **T**: Result of first agent D and second agent C has a variable value in range **[5,15]**.

The matrix is generated each round with equal probability for each point in the S, T plane. This variable values are chosen **randomly**, assuring this way there is no intention to condition the agent's decisions.

By having **11 options** for each variable, we will have a **11 x 11 matrix** of possible combinations. We are going to use this matrix for representing the obtained results depending on variables values and will be explain deeply in the data representation section.

3.2.3 Players phenotypes

For having different strategies, we have defined **5 different phenotypes** of agents. Each phenotype will follow its own rules, what will determine its behaviour.

As we want to make a tool as real as possible, our goal is to assess whether individuals behave in a highly idiosyncratic manner or whether, on the contrary, there are only a few "phenotypes" by which all our experimental subjects can be classified. As our goal is to make a tool for embed game theory and PECS reference model, not to classify human beings, we have taken the phenotypes described previously in the reference article used for developing this tool.

Coming up next, we are going to show the possible phenotypes and its heatmap. The cooperation level is represented from 0 in blue to 1 in red:

- **Optimist**: The first phenotype, called *Optimist* **cooperates** wherever $\boldsymbol{T < R}$. By using this strategy, these subjects **aim** to obtain the **maximum payoff** without taking into account the likelihood that their counterpart will allow them to get it.

  In other words, when he feels that he can obtain a higher value that the one offered by cooperating, it will defect its opponent.

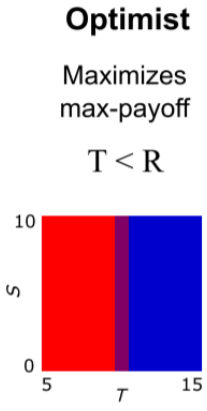

Figure 11: Optimist phenotype heatmap

- **Pessimist**: Conversely to the first, we label subjects in the second phenotype *Pessimists*, because they use a maximin principle to choose their actions, cooperating only when $\boldsymbol{S > P}$ to ensure a best worst-case scenario.

  As previous phenotype, its decisions can hardly be considered rational, and are related with different degrees of risk aversion.

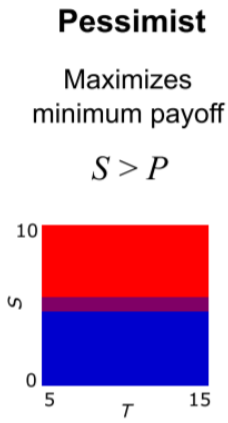

Figure 12: Pessimist phenotype heatmap

- **Envious**: The third phenotype is called *Envious*, because it exclusively cooperates in the upper-left triangle of the heatmap, when $\boldsymbol{(S - T) >= 0}$. Its name is given due to it takes decisions based in preventing their counterparts from receiving more payoff than themselves, even when, by doing so, they diminish their own potential payoff, which is far from being a rational decision.

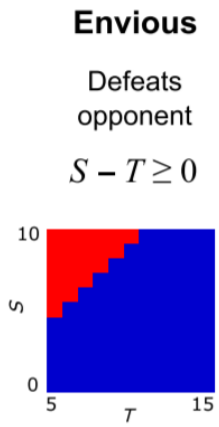

Figure 13: Envious phenotype heatmap

- **Trustful**: The four phenotype is called *Trustful*, because its unconditional faith in its opponents. It includes those players who cooperate always in almost every site of the *(S,T)* plane. This phenotype is the opposite to the previous one. Its behaviour can be associated with trust in partners behaving in a cooperative manner. They follow the formula $\boldsymbol{p(C) = 1}$.

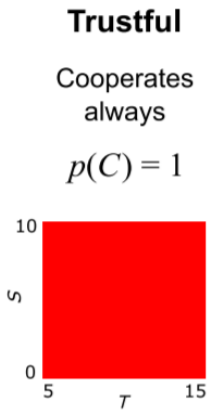

Figure 14: Trustful phenotype heatmap

- **Undefined**: The last group are the ones who cooperate in an approximately random manner, with a probability of 50% in any situation. For that reason, they are called *Undefined*. They follow the formula $\boldsymbol{p(C) = 0.5}$

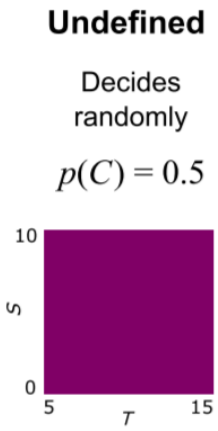

Figure 15: Undefined phenotype heatmap

3.3 Agents

Agents will play a key role in our system, as they are going to be the subjects of our further experiments over which we are going to study its behaviour by setting them a personality and a set of rules while seeing how they interact with the universe and other agents.

In this section we are going to talk about their different breeds, their internal variables that will condition part of its behaviour, the features that an agent holds and also how we are going to set up them at the start of each simulation.

3.3.1 Breeds

As we want to build a tool for holding simulations capable to represent real human behaviour, we need to make our agents as real as possible. As in real world not every person thinks or act the same, in our tool we need to have the ability of modelling different kind of agents.

For that reason, we have implemented **five** types of breeds, each one representing one of the previous mentioned phenotypes:

```
breed [optimists optimist]
breed [pessimists pessimist]
breed [enviouss envious]
breed [trustfuls trustful]
breed [undefineds undefined]
```

For distinguish them on the visual simulations, we have labeled each breed with a different colour. Also, they have the appearance of a person, because we have set *"person"* as default shape for all the breeds. This is the chosen breed representation:

- **Optimist** will be represented with colour **green**.
- **Pessimist** will be represented with colour **red**.
- **Envious** will be represented with colour **orange**.
- **Trustful** will be represented with colour **blue**.
- **Undefined** will be represented with colour **grey**.

The number of individuals of each breed is handled by five sliders on the *Netlogo* interface tab. Each one of them is related to a system variable that is used at the time of agent creation, during setup method.

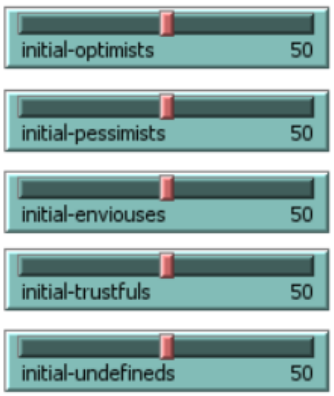

Figure 16: Slider variable number of individuals

3.3.2 Internal variables

Besides the breed, each agent have internal variables. These variables are needed to store data about the games that they are playing, to have knowledge against which other agents it has played during the current round or even to modelling emotions or cognition as we will see in the next chapter.

For this base implementation where we want to reproduce a reduced set of personalities without having any external input that can affect to its behaviour, we are going to use only one variable called ***games-current-tick***, which will store the id of our rivals during a certain round to assure that we are only playing against same agent once, at most, every round, to **avoid data duplication**. This agent variable is reset at the end of every round, and only works as a constraint for limiting games between same agents.

3.3.3 Setup

For setting up the agents before each experiment, we have implemented a method that holds all needed initialization of variables, breeds and even though, agents creation.

In this method, we define the five breeds available in our tool. Also, we assign to each one of them the shape of person for a better understanding during the simulation on the interface tab. Once we have defined the breeds, we define how many individuals we are going to create for each one, based on the previous slider variable introduced.

For each breed, we set up the previous assigned color, as well as initialize its list of agents whom has player in current round.

3.4 System architecture

All the reported information about our tool holds over a defined architecture defined in *Netlogo* framework. As we know that this implementation is just for setting up the structure for our finale implementation, we had in mind potential scale in its design, but implemented just our initial purpose.

We are going to explain everything on detail in next sections, but here there is an image of the tab interface of our tool.

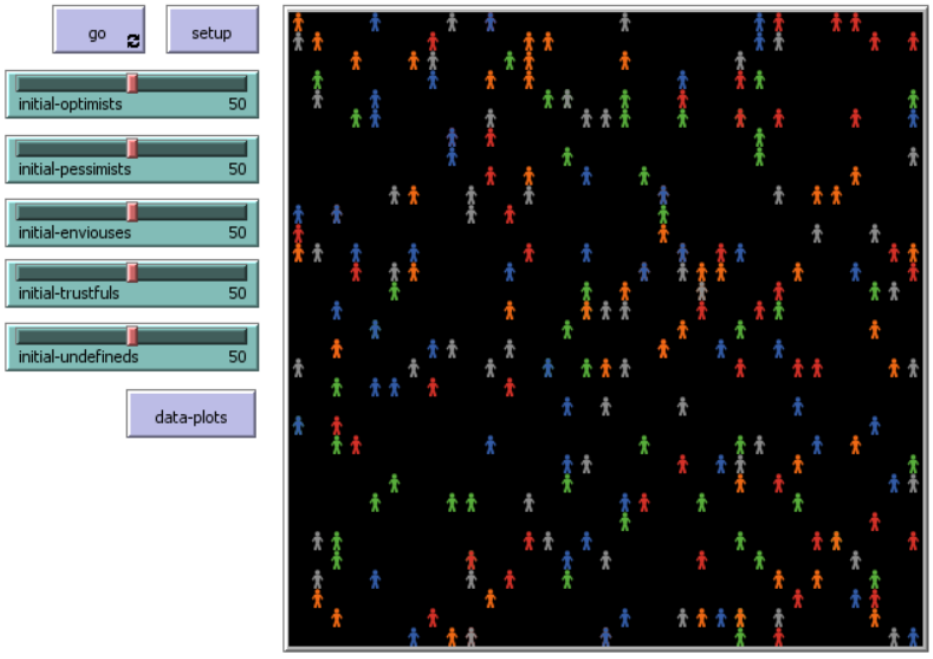

Figure 17: Netlogo interface of our tool initial approach

3.4.1 Constants and variables

The system, as responsible of data result report, must hold all the data that it is created during a simulation. Besides that, it also has to store the constants values needed for some methods during the simulation, and variable values that are changing on each round or growing up.

Following there are reported all constant and variable values used for develop the simulations.

Constants

- **payoffs**: This is where there are stored the values of the payoff matrix. Even though its values on the secondary diagonal are variable, we catalog it as a constant. It is checked after each game for determine the score of each agent.
- **C**: Here there is store the value of cooperate. It is used as index for accessing into a matrix specific value.
- **D**: Here there is store the value of defect. It is used as index for accessing into a matrix specific value.
- **R**: Here there is store the value of the action of both cooperation. It is used for check conditions of the agents depending on its breed and make a decision.
- **P**: Here there is store the value of the action of both defection. It is used for check conditions of the agents depending on its breed and make a decision.

Variables

As a simulation is a changing phenom, we need to have several variables for capturing all the events. This is a crucial part because it is not only needed for developing the simulation properly, but also it is needed to hold the results to be able of representing later and reach conclusions or discover phenoms.

These are the variables that handle all these data on this first approach:

- **S**: Variable that holds the random-generated every round value of the action where agent1 cooperates and agent2 defects.
- **T**: Variable that holds the random-generated every round value of the action where agent1 defects and agent2 cooperates.
- **num-games-per-round**: Variable that determines how many games are done on each round for every agent.

- **res-sum**: Matrix of 11x11 that holds the cooperation factor for a given values of S and T. Here it is stored the result of a game by using as indexes of the matrix S and T. This is used later for generate the heatmaps. There is on array per breed and the results of each personality are stored in its matrix.
- **res-times**: Similar to *res-sum*. It has the same sizes, but instead of store the result, it is stored the times that it happens for computing average once it is requested the data. Also, there is a matrix for each breed.

3.4.2 Checks and setup

During the setup phase, started when pressed setup button, our tool have some initialization needed for the proper development of the further simulation. Inside this setup method, we are also doing some initial checks needed before start a simulation.

The first action done is to reset the tick counter that are associated to each iteration of the simulation. After that, there are also cleared all the turtles.

Next methods executed inside setup are the system checks. In this part of the tool, we only have one active check that count all initial agents distributed into different breeds for assuring that we are not over passing the maximum agents number handled by the program.

Once we have assured the feasibility of the simulation, we set up the agents as we have explained on agent's section, and also we need to set up the system.

In the **setup-system** method we set the values of C to 1 and D to 0. Also, we generate the first two random values of S and T, and we use it to initialize **payoffs** matrix with them and also with the initialized values of $R = 10$ and $P = 5$. Finally, we initialize the five 11x11 matrix (one for each breed) that store the results of the game for S and T values. Initially, these matrix have all its positions equal to 0. Also the number of times matrices (*res-times*) are equally initialized too.

3.4.3 Methods

We are going to show the **four** methods that will handle the agents fate over the simulation, showing its pseudo-code of the implementation and explaining its behaviour. This four methods are inside a general method called **go**, which is represented in the interface tab of *Netlogo* as a button. Once the button is pressed, as it has marked the option *forever*, it will repeat it forever or until we push the button again.

action-rstp

This method is called inside the next method *play*. For each agent, and depending on its breed and the values of S and T, it decides its final chosen action. This is the pseudo-code of the method:

Algorithm 1 Pseudo-code of method **action-rstp**

Input: $agentBreed$
  if $agentBreed = optimist$ **then**
    if $T < R$ **then**
      return C
    else
      return D
    end if
  else if $agentBreed = pessimist$ **then**
    if $S > P$ **then**
      return C
    else
      return D
    end if
  else if $agentBreed = envious$ **then**
    if $S - T \geq 0$ **then**
      return C
    else
      return D
    end if
  else if $agentBreed = trustful$ **then**
    return C
  else
    return $random(C, D)$
  end if

play

Here is where all the 'action' is done. In this first phase, we select for each agent a set of agents to play with in base of variable **num-games-per-round**. For each rival, we check that there are not repetitions, and in case of have it any, we select randomly another opponent.

After that, for each agent and rival, we compute its decision based on its personality. Also, we add to the list of **games-current-tick** the rival and do the same from the rival point-of-view.

Once we have both decisions, we update heatmap matrix of each breed involved in the game with the chosen action. Here is the pseudo code of the method:

Algorithm 2 Pseudo-code of method **play**

```
for all agents do
  rivals = getRandomRivals(num − games − per − round)
  for all rivals do
    agentAction = actionRSTP(agentBreed)
    rivalAction = actionRSTP(rivalBreed)
    agentGamesCurrentTick.add(rivalID)
    rivalGamesCurrentTick.add(agentID)
    resSumAgentBreed[S][T] = agentAction
    resSumRivalBreed[S][T] = rivalAction
  end for
end for
```

update-variables

This method is in charge of update system and agent variables needed for starting a new iteration of the simulation, and for update changing data in order to have reliable results on the next iteration. Here we have a snippet of its pseudo code:

Algorithm 3 Pseudo-code of method **update-variables**

```
T = random(5, 15)
S = random(0, 10)
payoffs[C][D] = S
payoffs[D][C] = T
for all agents do
  gamesCurrentTick = []
end for
```

plots-data

This method is in charge of report breeds matrix for creating the heatmaps and evaluate the results. It is attached to the button ***do-plots*** from the interface tab of *Netlogo* and returns both **res-sum-<breed>** and **res-times-<breed>** for each breed.

We decided to report it this way (pre-processed, without compute average) because we taught that could be interesting for our tool to report any times as we want intermediate results that do not compromise the data. Here is the pseudo code of the method

Algorithm 4 Pseudo-code of method **plots-data**

return $resOptimist, resPessimist, resEnvious, resTrustful, resUndefined$

3.4.4 Data representation

For data representation we have chosen **heatmaps** because it is easier to represent the data on a visual way, to see how it affects to the cooperation the different values that S and T can take for each breed.

As *Netlogo* outputs only *raw data*, we have implemented a ***Python*** script that handles the matrix and use it with the help of the ***matplotlib*** library to plot the heatmaps. This is a sample of a resulting heatmap:

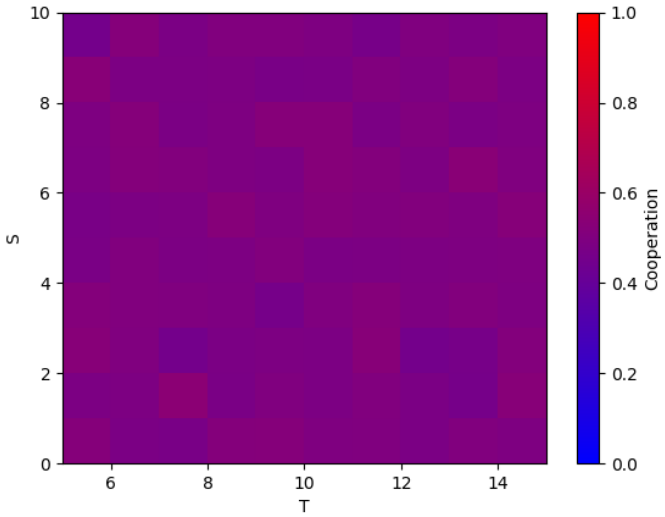

Figure 18: Heatmap sample

4. Final tool

4.1 PECS integration

The greatest challenge of our tool is to make feasible to have inside the same tool an agent-based modelling implementation with PECS reference model for taking part of the decision making embedded and game theory with PD as the way of generate options to the agents.

For that purpose, we need to **integrate** inside our tool the **PECS** reference model. PECS reference model is composed of four components: **P**hysical, **E**motional, **C**ognitive and **S**ocial status. We need to develop a module for each one of the four components and embed it to the agent for checking if they modify its behaviour to prove the feasibility of our tool

4.1.1 Physic and Social Status

For including these two modules for developing PECS model, we need to bring the capability of **movement** and **social interaction** to our agents. In this section we are going to explain how did we manage to bring to agents these two abilities.

Physic

On our first approach, agents were static actors that get a fixed number of opponents with the only condition of not playing against the same agent during the same round.

For changing this, we have gave to the agents the ability of moving free over the whole universe. Now, instead of receiving on each round a set of opponents, the agent is making a movement over the map every round, and based on its position, he will check its **eight potential neighbors**. For each neighbor patch that contains an agent, it will be added to the rivals list for that round and our agent will compete against them.

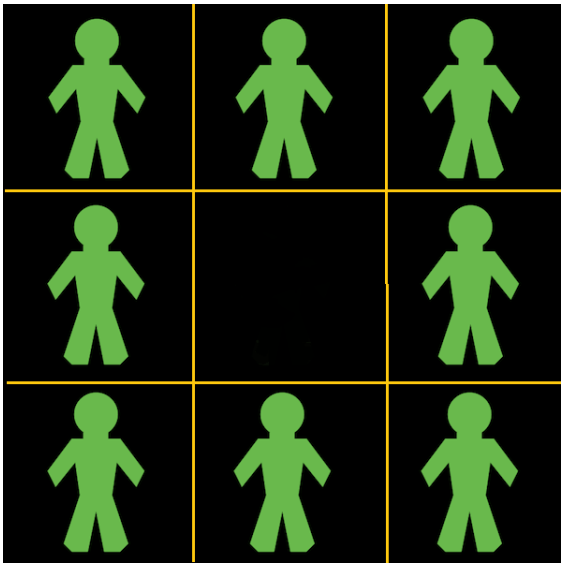

Figure 19: Potential opponents based on neighbors

Social status

The social status component was the easiest to embed in our tool. This is because it implies having relation with other human beings.

As in our first approach we were playing against other agents, we have fulfilled this condition, as the social interaction was covered yet by the act of playing to the PD against each others.

4.1.2 Emotion

The emotion component was implemented in our model in the form of revenge. We have added to the agents the capability of having 'memory', and now they are 'recording' everything that happens related to a game of the PD.

By building for each agent an array of size equal to the number of agents that are in our same universe, we are storing the last game that we had against each one of them. So then, the first time that we play against them, we will have no background of being potentially betrayed by them, and this component will not affect to the occurrence of the expectations. The same as if in the last game he did not betrayed us

But at the moment that the agent become betrayed, it store this information in the array of last games against agents, in the index of the agent, that he **defected**, and commit **betray**.

4.1.3 Cognition

The same way that we have used the agent's 'memory' for modelling emotion component of the PECS model, for the last remaining agent's capability we have done the same.

We have build an array with previous games done against agents for different values of T and S, and depending of the results that gave us best chances in the past, we are going to made our decision.

For the initial part of the cognition, as we do not have previous records, we are taking a **20%** of the total ticks of the simulation as **training**. We allow this 'licence' based on trial and test sets done in areas that requires a learning, and this learning (or training) phase should take about 10-20% of the total set, so we will assume too that the first 20% of the set of data will be our 'training'.

On the training stage we are going to made the decision in base to a formula that compute the average of scores for the current round of choosing to Cooperate or to Defect. If the condition is satisfied, we will choose to cooperate. Otherwise we will defect.

$$(R + S)/2) >= ((T + P)/2)$$

4.2 Tool changes

In order to make capable of make decisions not only by agent personality and embed PECS model, we have need to make some changes on our tool.

Besides the changes on the methods (that are explained in the next section), the variables and the interface, we have added a new check too based in the three new variables that we are going to introduce in this section, called ***theoretical-percentage, emotional-percentage and cognitional-percentage***. This checks guarantees that the sum of these three variables is not over 100 (total percentage).

4.2.1 Variables

We needed to grant agents with new set of variable for being capable of handling the previous commented abilities. These are these variables:

- **payment**: This arrays stores the results of all scores in games done by an agent.
- **payment-option-c**: In this matrix we store the score results where we have cooperated for each value of S and T.

- **payment-option-d**: In this matrix we store the score results where we have defected for each value of S and T.
- **theoretical-percentage**: Percentage (from 0 to 100) of the weight in the simulation of 'theoretical' result.
- **emotional-percentage**: Percentage (from 0 to 100) of the weight in the simulation of the emotional result.
- **cognitional-percentage**: Percentage (from 0 to 100) of the weight in the simulation of cognitional result.

4.2.2 Interface

As we have commented, also the interface suffered some changes due to the introduction of the PECS model. We had to add the capability of tuning the weight of each component that reach to a final action for deciding our final action. We have model it in a slider way to easier its modification:

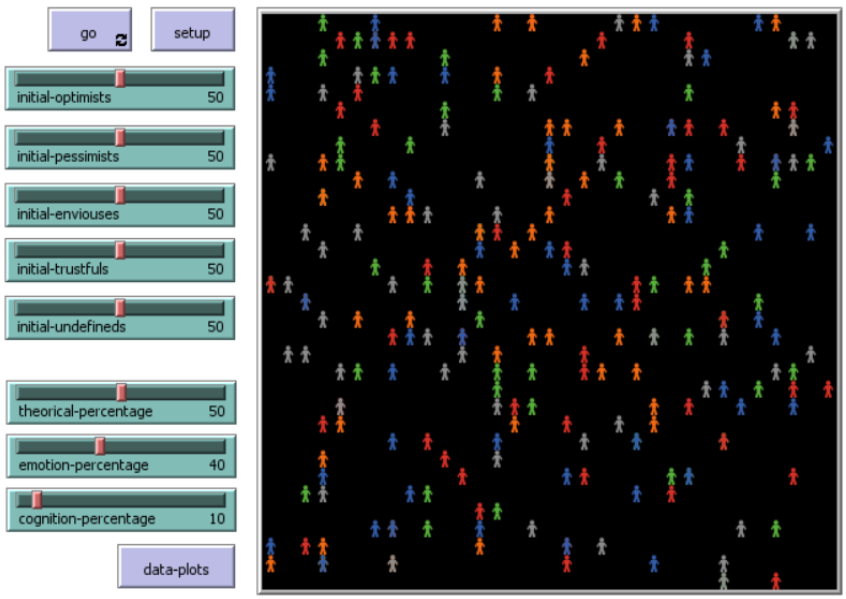

Figure 20: Final interface of our tool

4.3 Final methods

After building the whole tool as we wanted since the beginning of this project, here are the final state of the methods needed to develop the whole simulation accomplishing the previous mentioned behaviour.

4.3.1 Move

This is the first method executed during a simulation. It only has the purpose of for each turtle, generate a random value between 0 and 359 which will be its *heading*, and after choosing its heading, it set to the turtle the action of **forward 1**, which makes current turtle to move forward 1 unit of space in our universe.

This is the pseudo code of the method:

Algorithm 5 Pseudo-code of method **move**

```
for all agents do
    heading = random(0, 359)
    setForwardMovement(agent, heading, 1)
end for
```

4.3.2 Possible actions

Now, instead of acting based only into one component (the theoretical, which creates the personality), we have to choose between three by using the percentages for weighting more or less each of them.

For doing this, we are obtaining the results of each mode individually and in a independent way of its further use.

action-emotion

This method was the easiest to implement, as we only need to store the games done against all the agents and evaluate if we have been betrayed on the last game or not, or assume that we do not have been betrayed if it is the first time that we play (initialized as we have cooperated in a hypothetical first game). Here you can find its pseudo code:

Algorithm 6 Pseudo-code of method **action-emotion**

Input: $agentLastGame, rivalID$
  if $agentLastGame[rivalID] = D$ **then**
    return D
    return D
  else
    return C
  end if

action-cognition

For the cognition ability we have made more structural changes to our tool due to the necessity of more structures for storing the needed data, but as we had in mind the embedding of this component we made scalable our tool. Here is the pseudo code of the method:

Algorithm 7 Pseudo-code of method **action-cognition**

Input: $agentLastGamesC, agentLastGamesD$
  if $ticks/totalTicks < 0.2$ **then**
    if $(R + S)/2) >= ((T + P)/2)$ **then**
      return C
    else
      return D
    end if
  else
    $avgChoicesC = avgResult(agentLastGamesC[S][T])$
    $avgChoicesD = avgResult(agentLastGamesD[S][T])$
    if $avgChoicesC >= avgChoicesD$ **then**
      return C
    else
      return D
    end if
  end if

action-rstp

This method remains equal from our first approach. We will refresh the procedure by putting its pseudo code.

Algorithm 8 Pseudo-code of method **action-rstp**

Input: $agentBreed$
  if $agentBreed = optimist$ **then**
    if $T < R$ **then**
      return C
    else
      return D
    end if
  else if $agentBreed = pessimist$ **then**
    if $S > P$ **then**
      return C
    else
      return D
    end if
  else if $agentBreed = envious$ **then**
    if $S - T \geq 0$ **then**
      return C
    else
      return D
    end if
  else if $agentBreed = trustful$ **then**
    return C
  else
    return $random(C, D)$
  end if

action-decision

On this method we are only weighting by using weight variables of each component and reaching a consensus between the three decisions by computing a value between 0 and 1. If the computed value is greater or equal to 0.5, it will be consider at 1. Otherwise, it will consider as 0. Here is the pseudo code of the method:

Algorithm 9 Pseudo-code of method **action-decision**

Input: $emotion, cognition, theoretical$
  return $resultingAction = round(emotion * emotionPercentage + cognition * cognitionPercentage + theoretical * theoreticalPercentage)$

4.3.3 Play

The ***play*** method also has been modified to make it capable of develop the previous mentioned tasks. This is the final method pseudo code:

Algorithm 10 Pseudo-code of method **play**

```
for all agents do
  rivals = agent.getPotentialNeighbors()
  for all rivals do
    agentActionRSTP = actionRSTP(agentBreed)
    agentActionE = actionEmotion(agentLastGame, rivalID)
    agentActionC = actionCognition(agentLastGamesC, agentLastGamesD)
    rivalActionRSTP = actionRSTP(rivalBreed)
    rivalActionE = actionEmotion(rivalLastGame, agentID)
    rivalActionC = actionCognition(rivalLastGamesC, rivalLastGamesD)
    agentDecision = actionDecision(agentActionRSTP, agentActionE, agentActionC)
    rivalDecision = actionDecision(rivalActionRSTP, rivalActionE, rivalActionC)
    agentGamesCurrentTick.add(rivalID)
    rivalGamesCurrentTick.add(agentID)
    resSumAgentBreed[S][T] = agentDecision
    resSumRivalBreed[S][T] = rivalDecision
  end for
end for
```

4.3.4 Update variables and report data

For the update of the variables we do not have suffered changes. Neither for the data reporting method. These are the previous mentioned pseudo codes of the methods:

Algorithm 11 Pseudo-code of method **update-variables**

```
T = random(5, 15)
S = random(0, 10)
payoffs[C][D] = S
payoffs[D][C] = T
for all agents do
  gamesCurrentTick = []
end for
```

Algorithm 12 Pseudo-code of method **plots-data**

```
return resOptimist, resPessimist, resEnvious, resTrustful, resUndefined
```

5. Experimentation and results

Once the project development has been reported, it is time to make experiments with our tool and check the obtained results in order to evaluate the quality of the project.

As our goal is not to achieve results that reach into discover patterns on human behaviour, but create a tool capable of integrate game theory, PECS reference model and a given theoretical personality for being used as a starting point for develop further experiments, we have not focus our experiments on setting up the tool and reach conclusions about human behaviour.

Our experiments have been focused in demonstrate the functionality of the whole system with all different parts embedded working properly. For that reason, we have done **four** experiments.

For each one of the experiments, we are going to run the simulation during **1000 ticks**, as we have tried with several numbers of ticks for checking when we have relevant data in all of the positions of the heatmap, and also when the data results are stable, and we have observed that it happens after 900 ticks, so we have added around a 10% of contingency for assuring that we are covered in corner cases.

5.1 Theoretical experiment

We have denominated to the first experiment as the **theoretical experiment**. With this experiment we want to isolate the model for only have the results based on the personality of the agent. For that reason, we are going to set to **zero** the **emotion** and **cognition** percentages.

This is the interface setup for the experiment:

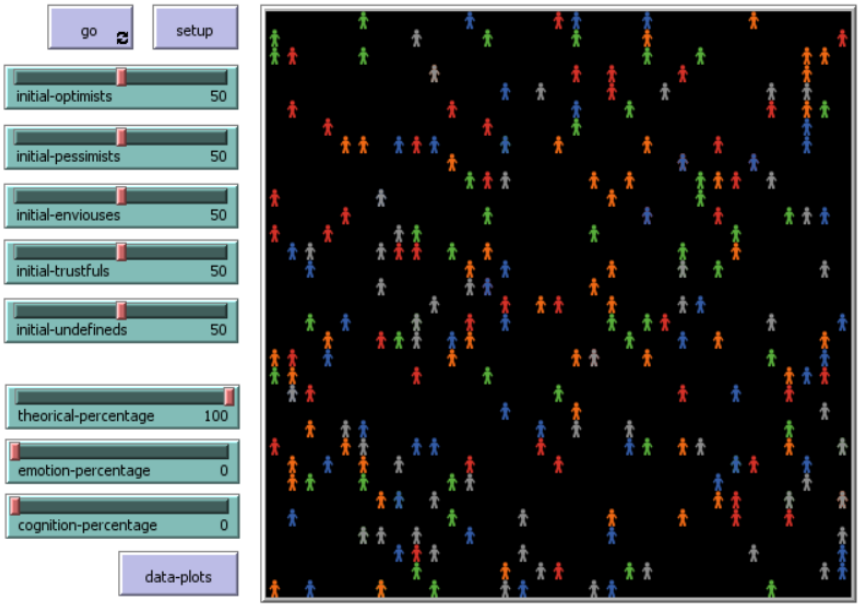

Figure 21: Interface setup for first experiment

After setting up properly and run the simulation over 1000 ticks, we have obtained the following results:

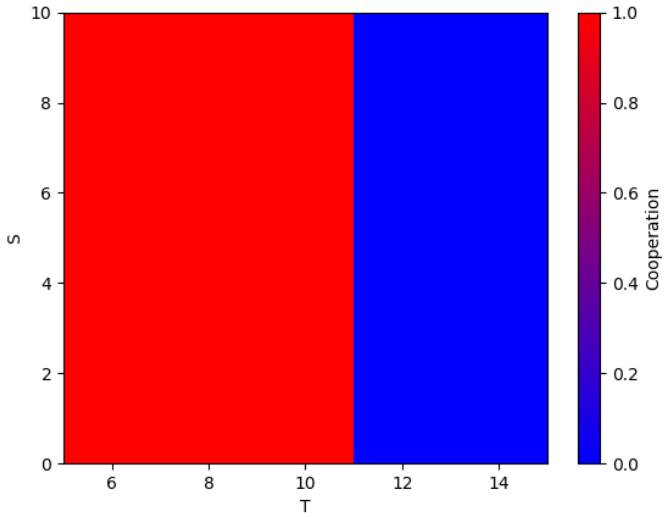

Figure 22: Heatmap of optimist breed

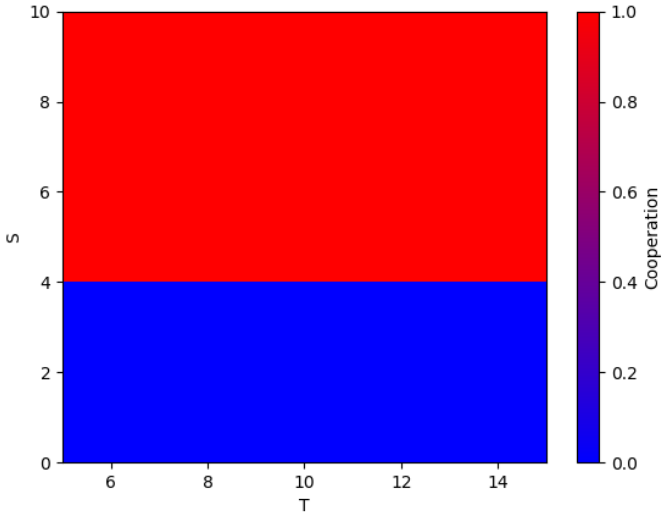

Figure 23: Heatmap of pessimist breed

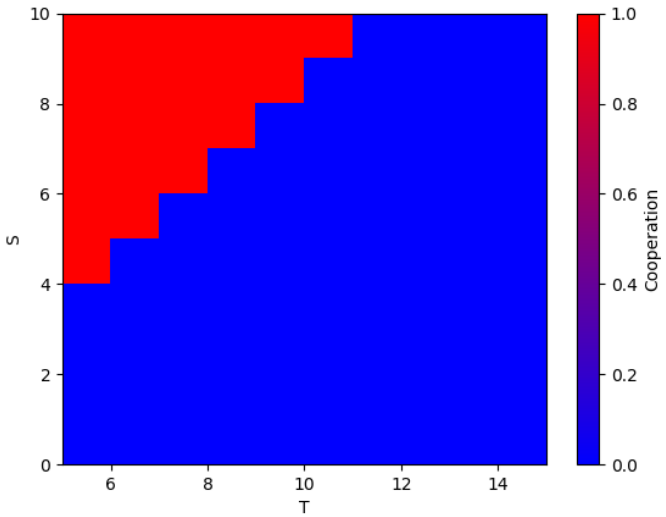

Figure 24: Heatmap of envious breed

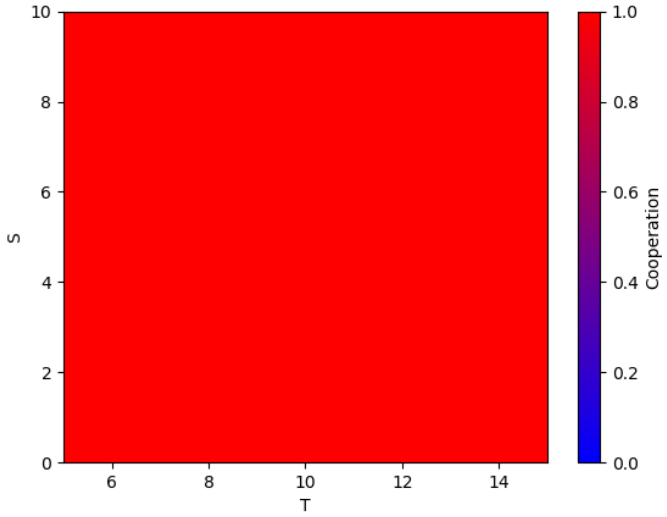

Figure 25: Heatmap of trustful breed

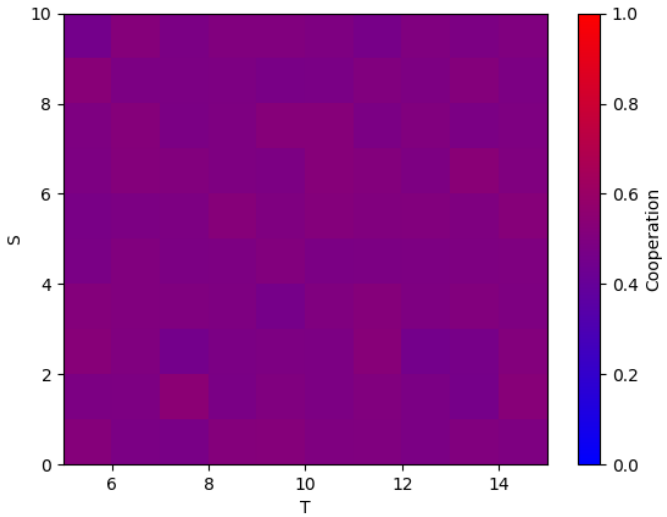

Figure 26: Heatmap of undefined breed

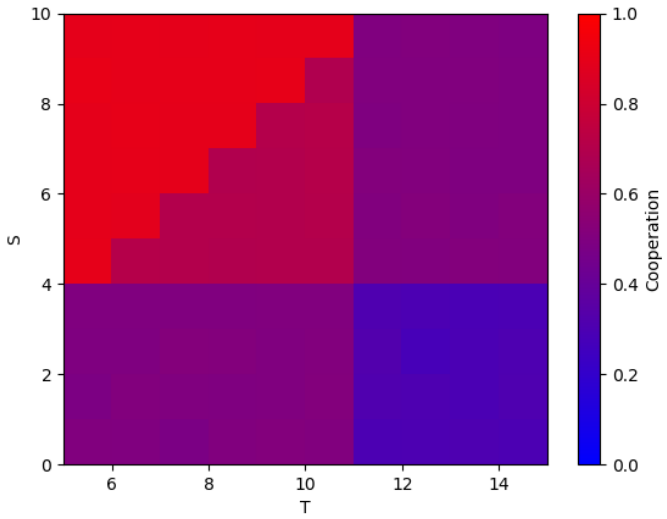

Figure 27: Heatmap of the aggregation of all breeds results

As we can observe from previous heatmaps, we are obtaining the **same results** that the **theoretical** ones showed during the explanation of the project on the initial architecture chapter.

Even though, we want to notice that there are not 'exactly' the same, but it is the expected behaviour. We can appreciate in the ***Optimist, Pessimist*** and consequently in ***Aggregation*** heatmaps that are slightly different in the border between full cooperation and full no cooperation.

This is happening because we are taking **strictly** the formulas as are reported in the article, and we are **no taking into account** the cases were $T = R$ for the *Optimist* and neither the $S = P$ case for the *Pessimist*. This two cases also affect to the final 'pattern' of the *Aggregation*, and so the difference too. We can observe too that is not happening on ***Envious*** heatmap because on its formula it is contemplated the equal case.

5.2 Irrational experiment

We have called this second experiment as the **Irrational experiment**. The name came from the fact that we are only using two of the three components that we had implemented capable of condition an act: ***Theoretical*** and ***Emotional***.

In this experiment we have set up at **50/50** the percentage of **theoretical** and emotion. The intention of this experiment is to check if the *Emotion* component is working properly, if it is capable of adding 'noise' to its behaviour, which will alter its personality.

This is the interface initial setup for the experiment:

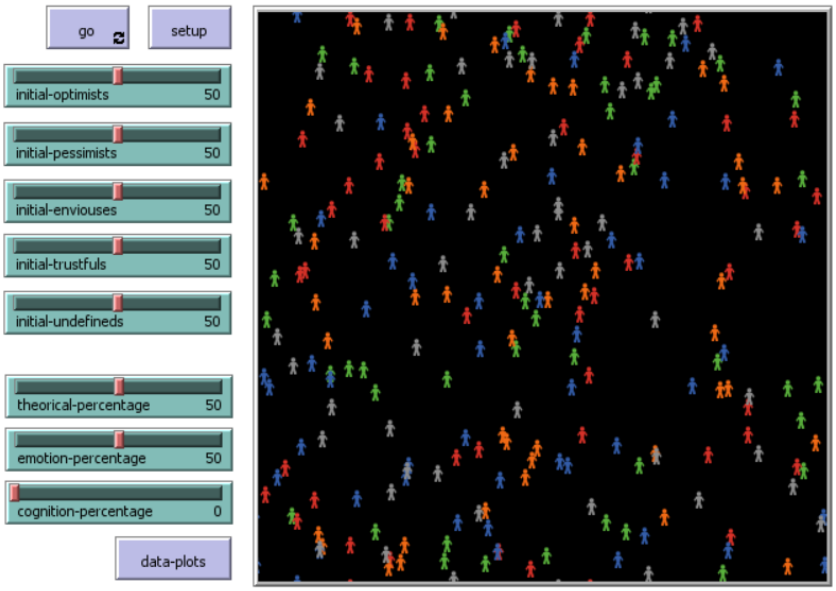

Figure 28: Interface setup for second experiment

After setting up properly and run the simulation again over 1000 ticks, we have obtained the following results:

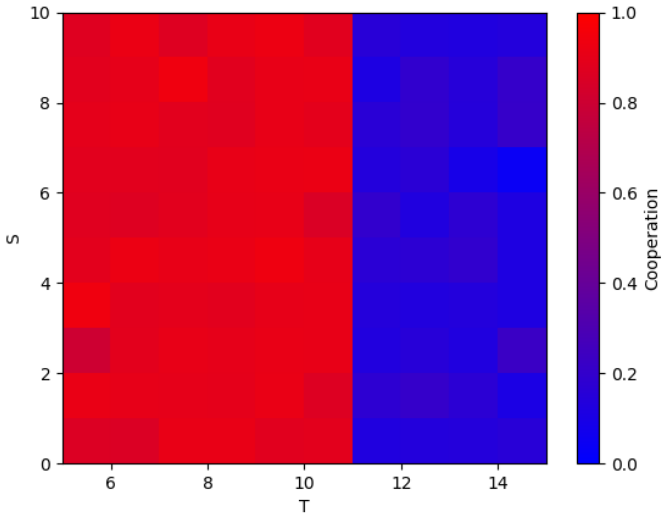

Figure 29: Heatmap of optimist breed

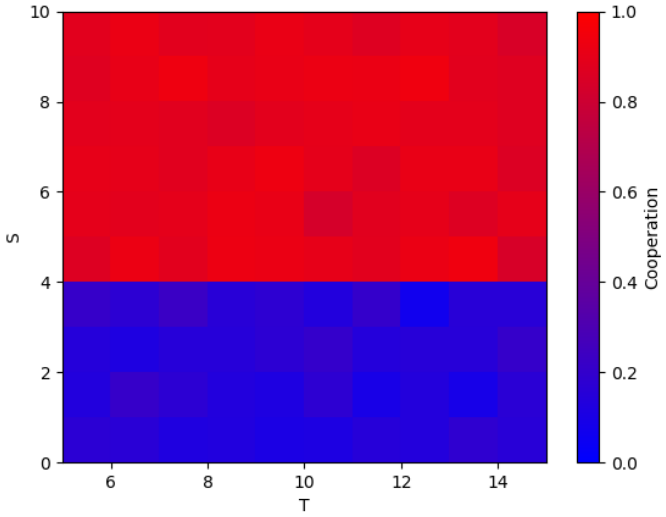

Figure 30: Heatmap of pessimist breed

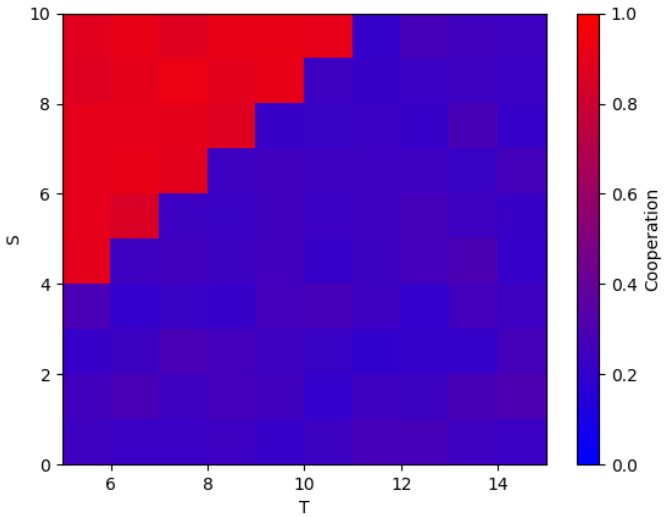

Figure 31: Heatmap of envious breed

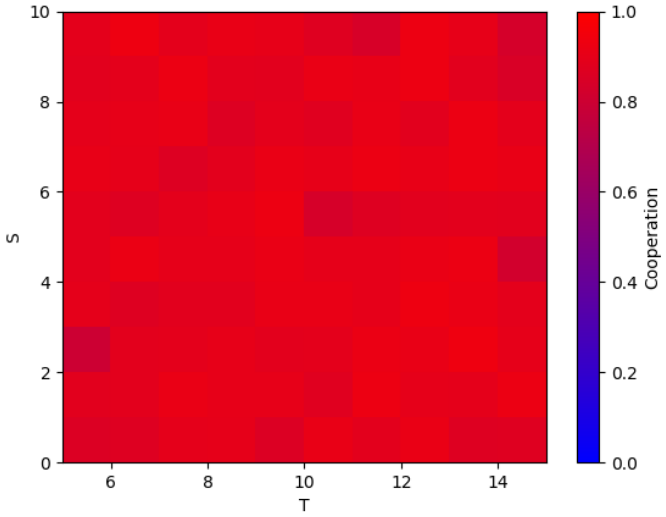

Figure 32: Heatmap of trustful breed

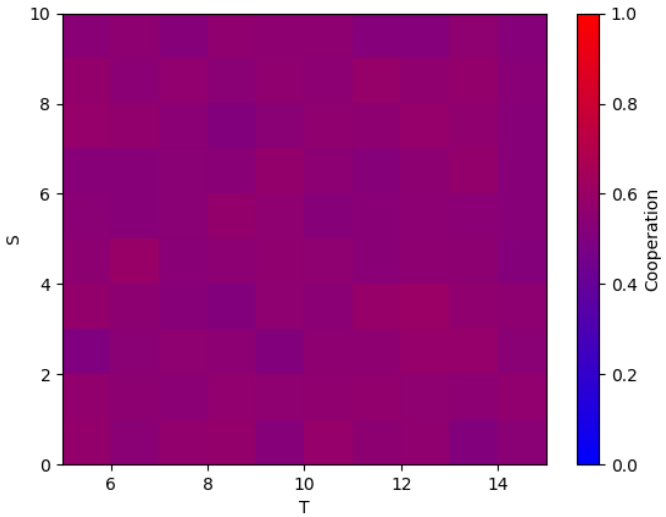

Figure 33: Heatmap of undefined breed

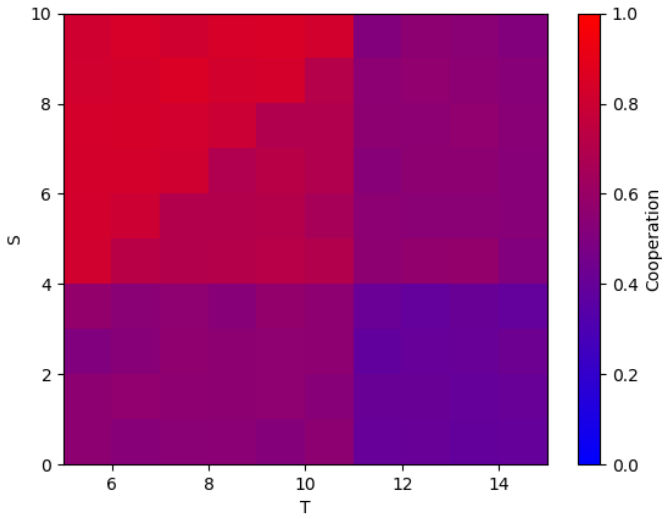

Figure 34: Heatmap of the aggregation of all breeds results

After show the different personalities heatmaps, we can observe that the ***Emotional*** component, as we expected, is adding noise to the agent's personalities. The intention of not get all the decision based on the emotion component but be equally balanced with the theoretical, which is the responsible of defining the personality, is precisely the main reason. We wanted to have sufficiently marked the personality, to check easily if it is affecting over it.

In general terms, the ***Emotional*** component is ***reducing the cooperation level*** mostly in all the personalities. The only exception can be the ***Trustful***, which seems less affected to the emotion noise, even though it has some areas with lesser cooperation.

Also, we can see in the *Aggregation* heatmap that the colors are more 'purple', what it means that in general, the impact of the emotion component is noticed, and it seems to decrease the cooperation. It is also true that as we are assuming at the initial game that the agent was not betrayed in the past, it is affecting to the 'blue' areas, which are the zones of less or no cooperation, which some cells of the heatmap are slightly clearer that in the theoretical, which means noise in form of cooperation.

5.3 Rational experiment

This third experiment is the the antithesis of the previous one. Instead of deactivating the ***Cognition*** component, we are balancing with the same weight as the *Theoretical*, while we are raising to zero the impact of the *Emotional* component. That is the reason why we have called to this experiment the **Rational experiment**, because it is pure cognition and agent's personality, inhibiting the emotion component.

This is the interface setup for the experiment, where we can see the 50/50 percentage balanced between ***Theoretical*** and ***Cognition*** components:

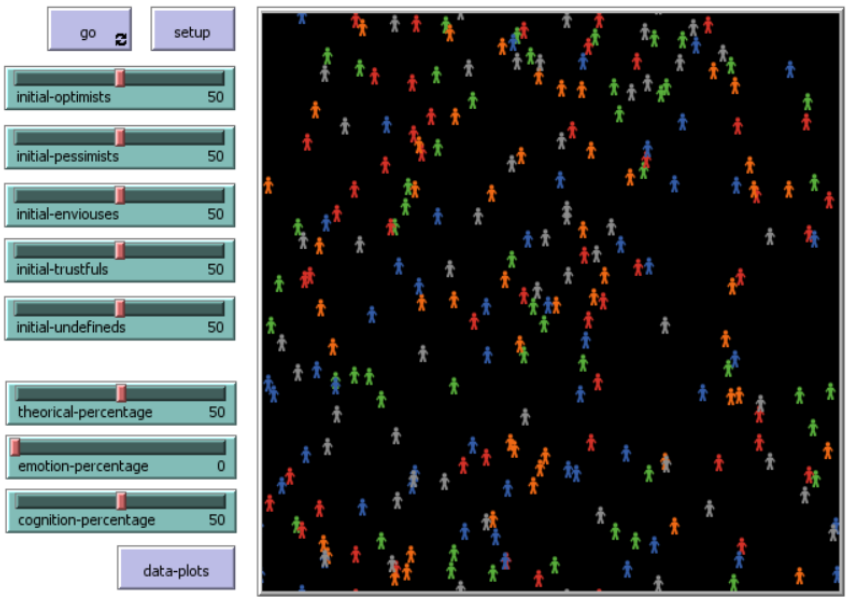

Figure 35: Interface setup for third experiment

Again, after running the simulation for 1000 ticks, these are the obtained results:

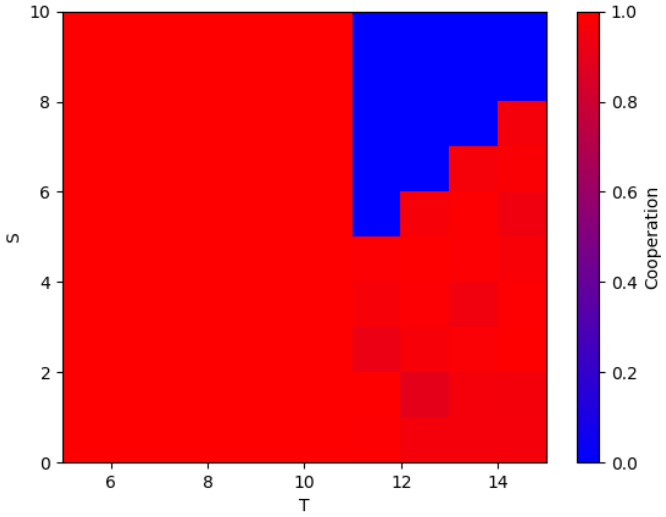

Figure 36: Heatmap of optimist breed

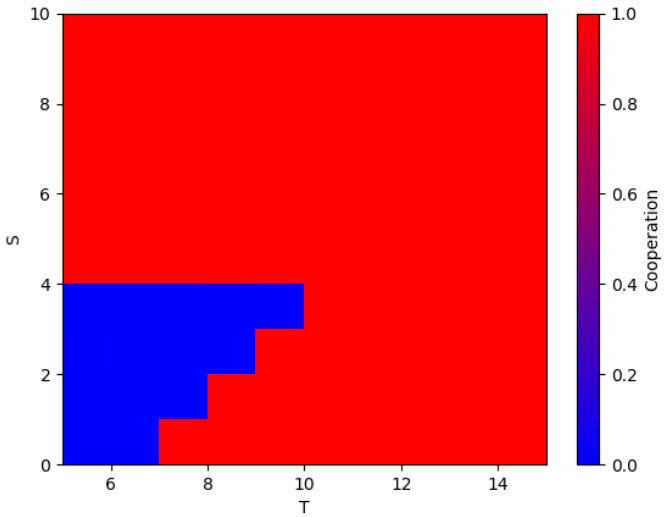

Figure 37: Heatmap of pessimist breed

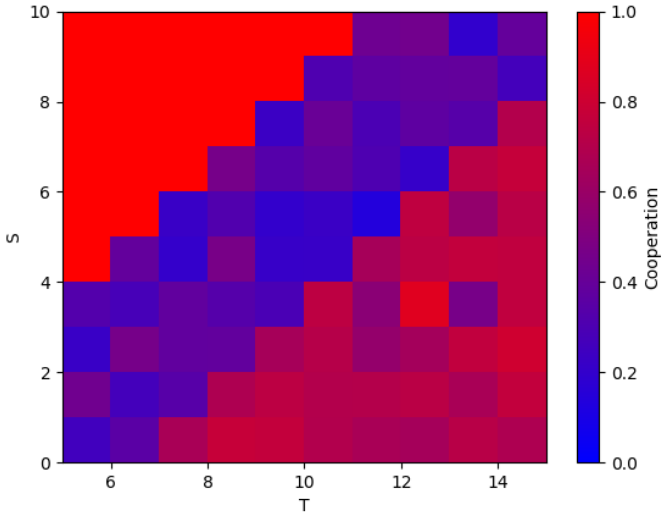

Figure 38: Heatmap of envious breed

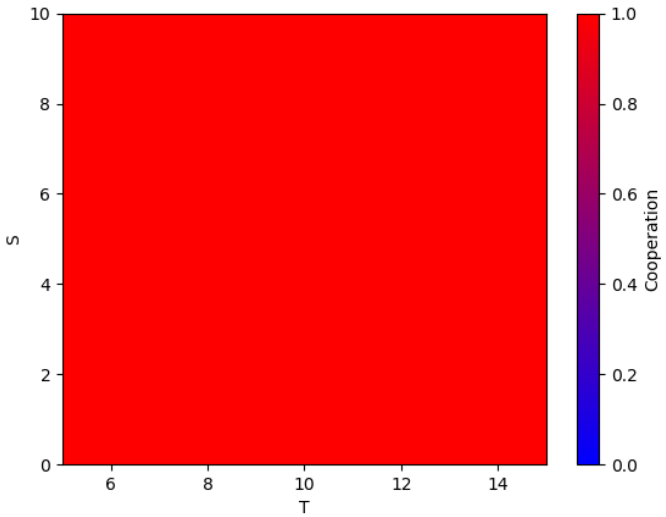

Figure 39: Heatmap of trustful breed

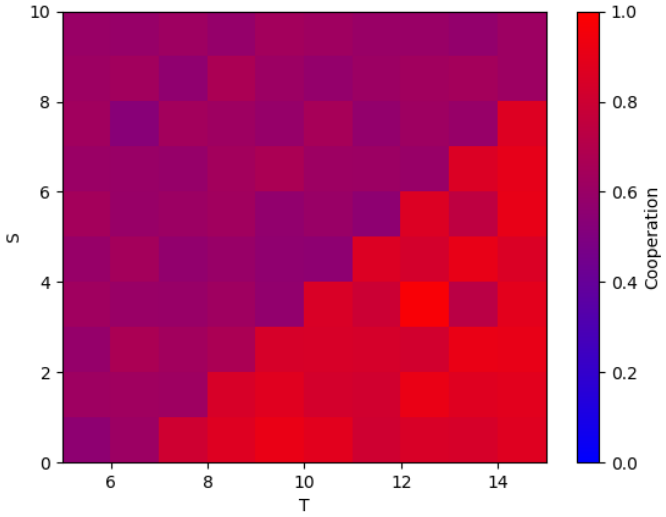

Figure 40: Heatmap of undefined breed

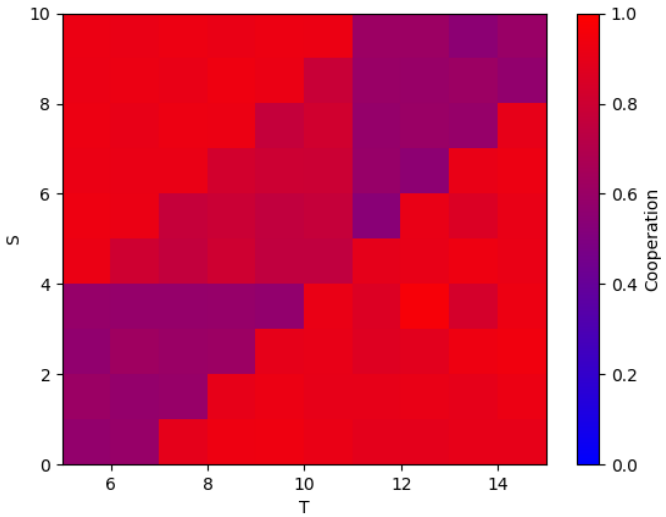

Figure 41: Heatmap of the aggregation of all breeds results

After showing the ration experiment results, we can see that cognition is affecting more clearly than emotion. In general, cognition had a big impact over all the personalities, with the exception of the ***Trustful*** personality, which is a singularity that clearly seems to be reinforced.

The impact over the other personalities **increases** considerably the cooperation levels. We think that this happens due to the impact of the **initial training phase** done during the 20% of the ticks, at the beginning of the experiment, to feed the memory with games for later, having enough knowledge to decide whether cooperate or not, based on its experience.

5.4 Balanced experiment

For this fourth and last experiment, we wanted to test all the components available for modifying the agent's behaviour available on our tool. As we want to remain the characteristicals of each personality, again we are going to set up the **half** of the **decision** based on the ***Theoretical***, and an equal proportion of **25%** on the other two components, the ***Emotional*** and ***Cognition***. As we are giving the same weight both of them, we have called it the ***Balanced experiment***.

This is the interface setup for the last experiment:

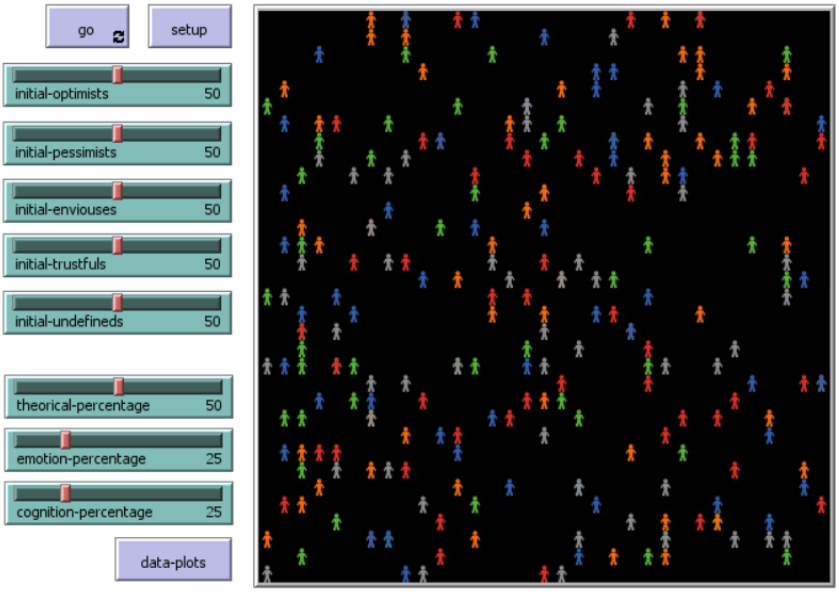

Figure 42: Interface setup for fourth experiment

After setting up properly and run the simulation over 1000 ticks, we have obtained the following results:

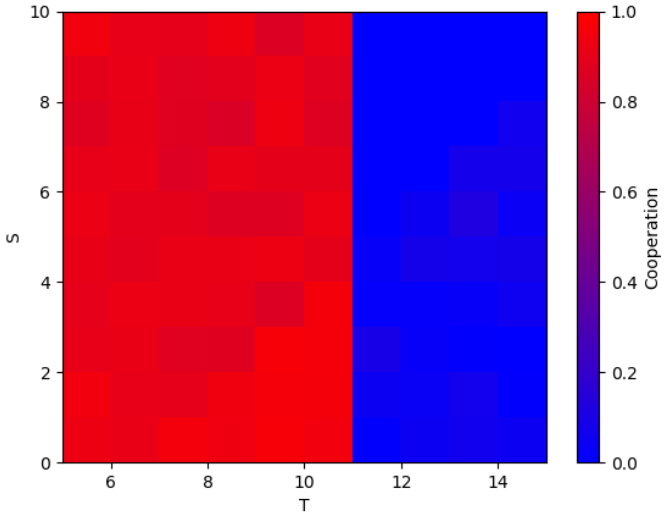

Figure 43: Heatmap of optimist breed

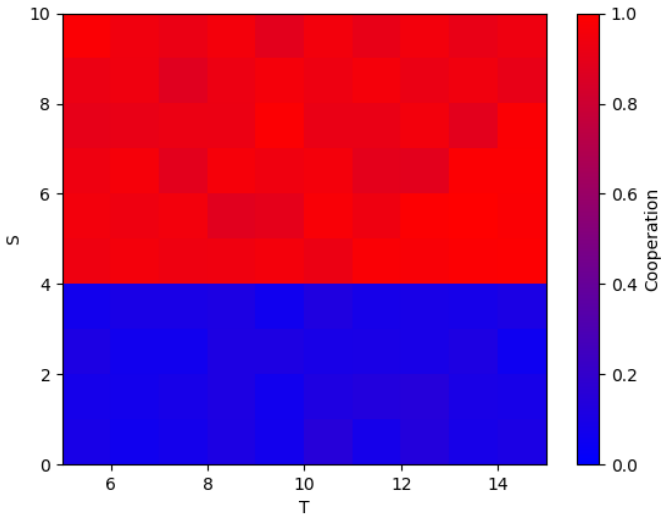

Figure 44: Heatmap of pessimist breed

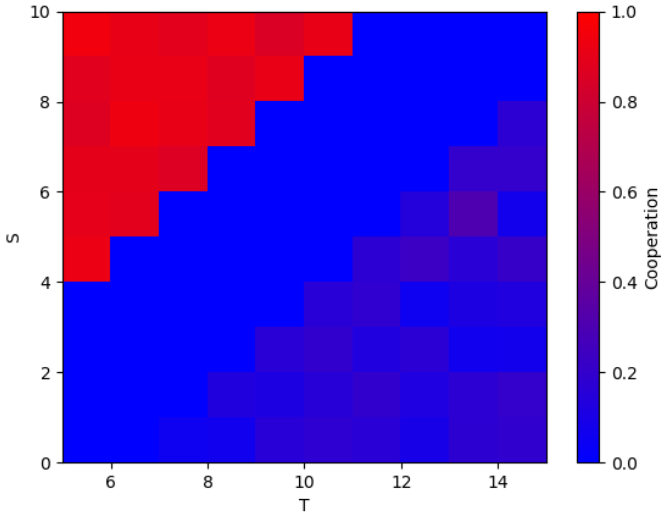

Figure 45: Heatmap of envious breed

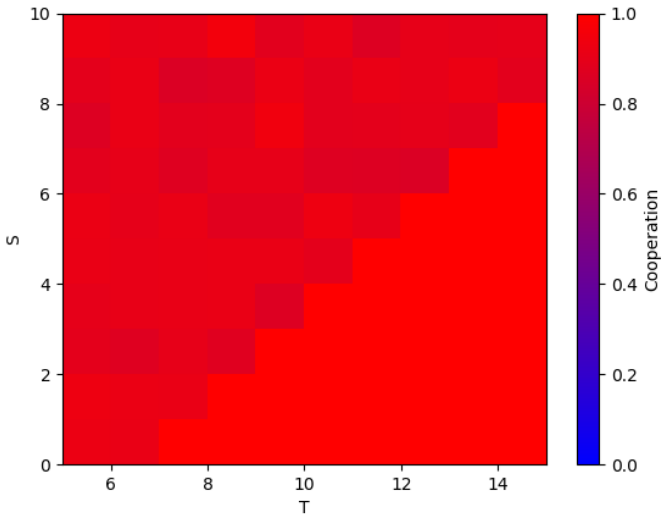

Figure 46: Heatmap of trustful breed

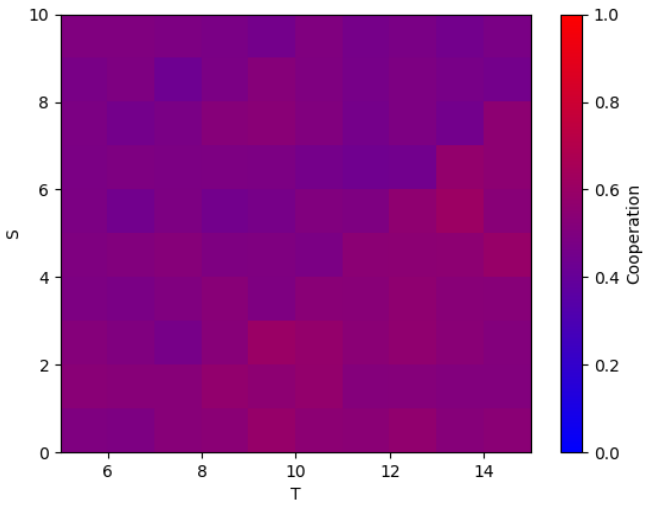

Figure 47: Heatmap of undefined breed

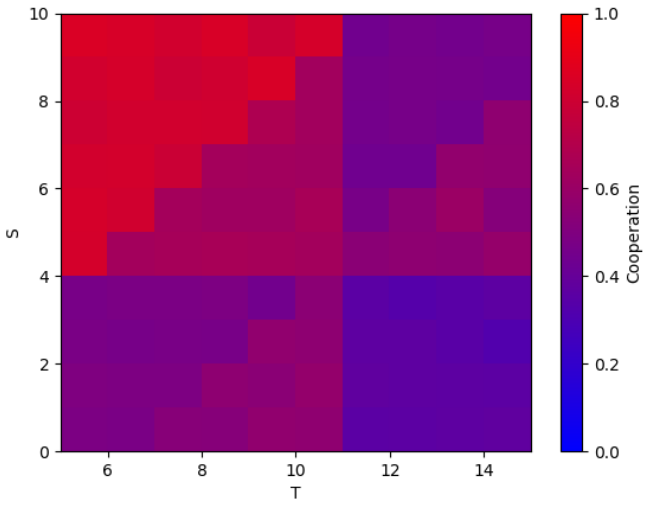

Figure 48: Heatmap of the aggregation of all breeds results

After this last experiment and having the background of the two previous one isolating the components, we can clearly observe a **mix** of the **Emotion** and **Cognition** components over each personality, fact that is telling us that the repeatability in the results of our tool is working fine, and so, the **components are impacting over agent's personality**.

6. Conclusions and Future Work

The **goal** of our project was to build a tool in *Netlogo* framework **capable of embed *game theory***, ***PECS reference model*** and ***agent-based modelling*** simulations that can be used in further works or investigation projects as a starting point.

By proving the feasibility of our tool by developing a set of experiments where we can observe that we have an **Agent-Based Modelling** running simulation where agents are **moving** freely over a defined universe and **meeting** opponents for **playing with** to the **Prisoner's Dilemma**, a well-known **games theory** paradox that implies cooperation between participants, with **different agent's personalities** that have the impact of the agent's **emotions**, based on previous experiences against its rival, and also the **cognition** impact in form of memory, having knowledge of all previous games played versus all met agents in our universe, we can conclude that **we have reached the initial goal**.

Even though we have accomplished our tool goal, there is still a lot of way to be done, because it is only the starting point of a large journey through investigation of ways for improving it.

The cognition and emotion 'formulas' have been taken from theoretical, are not empirical facts. As our point was not predict human behaviour, we were not focused in making several experiments, also involving real people and not only simulations, to tuning up the components, *Emotion* and *Cognition* in our case, for being as 'human-like' as possible. It could be a **good starting point for make a new research by using our tool and improving it in that way**.

Also, our **agents have a very strong root to its defined personalities**, which in fact was fed by us, with the idea of having results near of its personalities to can clearly check that the tool was modelling what it was designed for, but it will be interesting to change these root personalities and transform it into agent's variables, with several parameters that can **personalize** a starting personality as could be an *Optimistic* person, but with its own features that distort it.

It would be also interesting to **add more components besides the ones that *PECS* reference model have**. Further research can lead into that direction, by studying human beings and try to extract from them new pattern not know yet that can be out of the scope of the *PECS* model, which will make our tool easier to predict real human behaviour.

Bibliography

[1] Phys.org. *Human behavior is 93 percent predictable.* [Online] Available from: https://phys.org/news/2010-02-human-behavior-percent.html

[2] C. Song, PhD. 2010. *Yet, we have found that despite our heterogeneity, we are all almost equally predictable.* Boston: Northeastern University College of Science.

[3] M. Garry, R. Michael and I. Kirsch. 2012. *Suggestion, Cognition, and Behavior.* Association for Psychological Science.

[4] F. Klügl and A. Bazzan. 2012. *Agent-Based Modeling and Simulation.* Association for the Advancement of Artificial Intelligence.

[5] R. Myerson. 1997. *Game Theory: Analysis of Conflict.* Harvard University Press, Cambridge, Massachusetts, London, England.

[6] C. Urban. 2000. *PECS: A Reference Model for the Simulation of Multi-Agent Systems.* Universität Passau, Passau, Germany.

[7] M. Georgeff, B. Pell, M. Pollack, M. Tambe and M. Wooldridge. 1998. *The Belief-Desire-Intention Model of Agency.* Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages, Paris, France.

[8] J. Wright and K. Leyton-Brown. 1998. *Beyond Equilibrium: Predicting Human Behavior in Normal-Form Games.* Departament of Computer Science, University of British Columbia, Vancouver, Canada.

[9] C. Urban and Dr. B. Schmidt. 2001. *PECS – Agent-Based Modelling of Human Behaviour.* Universität Passau, Passau, Germany.

[10] Dr. B. Schmidt. 2002. *Modeling of Human Behavior. The PECS Reference Model.* Universität Passau, Passau, Germany.

[11] J. Poncela-Casanovas, M. Gutiérrez-Roig, C. García-Lázaro, J. Vicens, J. Gómez-Gardeñes, J. Perelló, Y. Moreno, J. Duch and A. Sánchez. 2016. *Humans display a reduced set of consistent behavioral phenotypes in dyadic games.* American Association for the Advancement of Science, Washington, USA.

[12] M. Georgeff, B. Pell, M. Pollack, M. Tambe and M. Wooldridge. 1998. *The Belief-Desire-Intention Model of Agency.* Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages, Paris, France.

[13] U. Wilensky. 1999. *Netlogo* [Online] Available from: https://ccl.northwestern.edu/netlogo/

[14] G. Nebehay. 2011. *Logo – Wikipedia.* [Online] Available from: https://en.wikipedia.org/wiki/Logo_(programming_language)

[15] E. Bonabeau. 2002. *Agent-based modeling: Methods and techniques for simulating human systems.* Proceedings of the National Academy of Sciences of the United States of America (PNAS), USA.

[16] Stanford Encyclopedia of Philosophy. *Game Theory.* [Online] Available from: https://plato.stanford.edu/entries/game-theory/

[17] International Encyclopedia of the Social Sciences. *Nash Equilibrium.* [Online] Available from: http://www.columbia.edu/~rs328/NashEquilibrium.pdf