Ατομική Διπλωματική Εργασία

ΑΝΤΙΜΕΤΩΠΙΣΗ ΤΟΥ ΑΥΤΟΕΛΕΓΧΟΥ ΣΑΝ ΔΕΞΙΟΤΗΤΑ ΣΕ ΕΝΑ ΥΠΟΛΟΓΙΣΤΙΚΟ ΜΟΝΤΕΛΟ

Γεωργίου Κυριάκος

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Επιβλέπων Καθηγητής

Χρίστος Χριστοδούλου

Ιούνιος 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Treating self-control as a skill in a computational model

Georgiou Kyriakos

Επιβλέπων Καθηγητής

Χρίστος Χριστοδούλου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Ιούνιος 2021

Acknowledgments

I would like to especially thank my supervisor, Dr Chris Christodoulou for his significant support and guidance throughout all the research stages. He taught me many useful things and I am grateful that I was given the opportunity to take part in such an interesting research topic. I also want to extend my thanks to my family and close friends. Their unwavering support was a key factor for me to stay focused and motivated throughout this period.

Abstract

This thesis suggests incorporation of theories that could affect positively or negatively the behavior of self-control in a computational model. These theories align with the belief that self-control should be viewed and treated as a skill, a virtue and as a muscle. Self-control is defined as the dilemma between a smaller sooner reward (SS) and a large later reward (LL) (Rachlin, 2000), that is experienced as an internal conflict between the higher (prefrontal cortex) and lower (limbic system) parts of the brain . The Prisoners Dilemma (PD) game is used to effectively simulate the nature of this conflict and the upper and lower parts of the brain are represented with two agents. The agents are engaging in the Iterated Prisoner's Dilemma (IPD) game, through a Q-Learning model with the goal of learning to cooperate and thus achieve self-control. The values of the IPD's payoff matrix T, R, P, S, define the reinforcement signals that the agents receive according to their collective actions during each round of the game. Each implemented concept was simulated in compliance with what the psychologists and neuroscientists suggest, indicating that attained results and the model are cognitively adequate. The self-control "strength model" (Muraven and Baumeister, 2000) was the basis on which we simulated the behavior as a body muscle. The depletion of self-control muscle (lack in self-control resource or strength) was amplified when high conflict payoff matrixes were used and it was restricted when the system was entering recovery states by alternating between a higher and a lower conflict payoff matrixes during the IPD game. Furthermore, an increase or decrease in self-control outcome was found when an initial self-control task (payoff matrix) was followed by subsequent tasks of similar or different response conflict, respectively. When extrinsic motivation (Berlyne, 1966) factors were presented during the IPD game there was an increase on self-control outcome, whether that factor was an external reward or an external player forcing the agents to periodically choose to cooperate. The final theory we incorporated was self-control virtue theory according to which self-regulation is defined by the individual standards and his/her ability reset those standards (Baumeister & Exline ,2001) while in the presence of negative emotions. The intensity of the experienced negative emotion (decreased R or increased T) during the IPD game defined how often the payoff matrix was required to be reset back to its initial-standard payoff values, in order for self-control to be achieved.

Contents

Acknowledgements…….....iii

Abstract……iv

Contents ……...v

Chapter 1 Introduction..... …..1

1.1 Introduction..... …..1

1.2 Thesis outline..... …..4

Chapter 2 Epistemological background

2.1 Self-control …….....5

2.1.1 The top-down model of self-control…….7

2.2 The Prisoner's Dilemma (PD) ……9

2.2.1 Iterated Prisoner's Dilemma (IPD) ……12

2.3 Self-control Strength model ……13

2.3.1 Muscle Recovery theory ……14

2.4 Self-control Motivators …….....14

2.5 Self-control Virtue …….....15

2.6 Reinforcement Learning ……..16

2.6.1 The Q-Learning Algorithm ……19

2.6.2 The ε-greedy policy ……20

Chapter 3 Design and implementation

3.1 Introduction ……21

3.2 Simulating Self-control Strength Model ……22

3.2.1 Muscle Recovery ……23

3.2.2. Subsequent Self-control Tasks ……..24

3.3 Simulating Extrinsic Motivation …….....25

3.4 Simulating Self-control as a Virtue …….....26

3.5 The Q-Learning agents ……..26

3.6 The Q-Learning model ……27

Chapter 4 Results and discussion

4.1 Introduction ……..31

4.1.1 Constant payoff matrix ……31

4.2 Simulating Self-control Strength model ……33

4.2.1Ego depletion ……33

4.2.1.1 Increasing the Total number of rounds of IPD ……..34

4.2.1.2 Increasing the Intensity of Internal conflict ……...36

4.2.2 Simulating Muscle Recovery ……37

4.2.3 Simulating Task switching ……41

4.2.3.1 Switching to Similar Intensity conflict tasks ……41

4.2.3.2 Switching to Different Intensity conflict tasks ……..43

4.2.4 Summary and discussion on Self-control Strength model ……45

4.3 Extrinsic Motivation ……..……46

4.3.1Simulating External monetary reward ……..47

4.3.2 Simulating External motivator …….....49

4.3.3 Summary and discussion on Extrinsic motivation ……..52

4.4 Simulating Self-control as a virtue ……..52

4.4.1Presence of Negative emotions ……53

4.4.2 Simulating Clear standards ……..55

4.4.2.1Decreasing the Reward payoff ……..56

4.4.2.2 Increasing the Temptation payoff ……58

4.4.2.3 Increasing Temptation and Decreasing Reward payoff ……60

4.4.3 Summary and discussion on self-control virtue theory ……62

Chapter 5 Conclusions

5.1 Overview and conclusions 63

5.2 Future work 66

References 68

Appendix A A-1

Appendix B B-1

Chapter 1

Introduction

1.1 Introduction ……1
1.2 Thesis outline ……4

1.1 Introduction

The purpose of this thesis is to examine theories that could affect positively or negatively the behavior of self-control. The theories are aligned with the belief that self-control should be viewed and treated as a skill, a virtue and as a muscle. Self-control refers to the capability to regulate one's emotions, thoughts, and behavior in the face of temptations and impulses. In cognitive psychology it is defined as the dilemma of choosing between a smaller sooner (SS) reward versus a large later (LL) reward (Rachlin ,2000). People experience everyday these types of dilemmas especially when they are considering changing an aspect of themselves. For instance, a person who committed to quitting smoking has to maintain control and resist the immediate sooner reward which is to light up a cigarette (SS reward), in order to achieve the long-term goal of improving his health (LL reward). Unfortunately, change is rarely aligned with choosing the easiest option (SS reward) at any given moment but it is mainly aligned with the choice of commitment (LL reward).

The dilemma described above is experienced in the brain as an internal conflict between the higher and lower parts of the brain. Neuroscientists confirmed this internal conflict by studying the activation of brain during the anticipation of long-term rewards and the exertion of self-control (Hare et al., 2009; McClure et al., 2004).The higher parts of the brain are described by the brain's prefrontal cortex which is responsible for the intellectual processed responses and could be viewed

as the “rational” brain . On the other hand, the lower parts of the brain are described by the brain’s limbic system that is responsible for the subconscious autopilot responses that are more aligned with our animal instincts and motivations. To simulate the conflict we deploy the general-sum game of the Prisoner’s Dilemma (Kavka, 1991) with two agents representing the lower and higher parts of the brain . Each agent is driven by his own motives and is focused on maximizing its own reward by defecting, however the best outcome for the organism and the group is only achieved in mutual cooperation. During the game, the agents have to choose between cooperating (C) or defecting (D) with each of them not knowing what the other one will choose to do. All the combinations of states in which the agents can result in are CC, CD, DC, DD. The state of self-control is represented by the CC state. The self-control model that is going to be used is similar to the one Georgiou (2015) and Nikodemou (2020) used on their theses. The model consists of two agents representing two parts of the brain that interact by engaging in the Iterated Prisoner’s Dilemma (IPD). The Q-Learning algorithm was used to train the agents and the desired outcome comes when both agents learn to cooperate (CC states) and therefore, achieve self-control. Previously, Banfield (2006) and Cleanthous (2010) , modelled self-control behaviour using neural networks and it was proven that the deployment of biological realistic spiking neural networks, was not necessary in order to successfully model self-control (Christodoulou et al., 2010)

The main part of the thesis is focused on reinforcing the idea that self-control should be treated as a muscle and should be trained as such in order to be improved. The psychology model aligned with this concept is the “self control strength model ”. According to this model, self-control resembles a muscle and the ability to exhibit self-control relies on a limited resource, or self-control strength, where all different self-control operations draw on that same resource (Muraven and Baumeister,2020).Before Cleanthous (2010) incorporate some of the ideas related to strength model and showed that exercising self-control similar to a body muscle helps resolving intense internal conflicts .The first method we will be considering is the muscle recovery theory . Assuming that self-control represents a muscle, incorporating recovery periods during the interaction of the agents in the IPD game could influence positively the desired outcome for the organism. Recovery period in our case is modeled by decreasing the level of conflict that the agents experience for a certain amount of rounds in the game. The next method, we will be using to decrease the negative consequences of self-control depletion is by applying the theory of extrinsic

motivation in our model. A person is extrinsically motivated if he performs an activity driven by an external reward that is not directly associated with the activity itself (Berlyne, 1966). An example is a student who studies driven by the desire to get good grades (external reward). To simulate the external motivator, we change the discount rate of the higher brain agent (limbic system) as soon as the effects of depletion appear during the IPD game. Decreased discount rate means that the organism gives greater value to the future reward (LL), which is a result of the external reward that showed up. Furthermore, in continuation with the self-control strength model theory we will be examining the outcome that comes when the system participates in subsequent self-control tasks. According to studies performed by Dewitte, Bruyneel and Geyskens (2009) self-control performance varies when the subsequent self-control task involved a different response conflict or a similar response conflict. Modeling the subsequent tasks is achieved by changing the value payoff of the system during the game. According to previous studies on the topic, it has been proven (Cleanthous,2010) that the level of conflict that the agents have during the IPD game is proportional to the difference between the Temptation to defect reward (T) and the Sucker's payoff reward (S). The level of conflict could also represent the level of difficulty of the task that the agents ought to achieve self-control (CC States). We will be using this approach in order to simulate the difference in conflict between the subsequent tasks.

Finally, the last position to take into account is the self-control virtue theory according to which self-regulation is analyzed into three main ingredients: standards, monitoring, and operations that alter the self. The ability of the individual to reset his standards while in the presence of negative emotions is an important factor determining whether self-control will be accomplished or not. In order to simulate the presence of negative emotions we will be using the same approach that Nikodemou (2020) used, where at the course of the game we will be increasing or decreasing periodically the T or R value respectively. The system will then periodically return to the relevant standards by setting the payoff value matrix back to the initial one.

Through the implementations we will be discussing whether and how our results are in alignment with what the Psychology and Neuroscience bibliography suggests on each self-control theory in order to prove the cognitive accuracy of the model.

1.2 Thesis outline

Chapter 1 is an introduction and Chapter 2 presents the epistemological background of self-control behavior in psychological and neurobiological terms. In this chapter we will look at the concept behind each theory of self-control and the definitions that the simulations are based on. First, we will examine the theory of self-control muscle and virtue as well as the findings that psychologists found that are been related to the theory. In continuation, we will explore the influence that intrinsic and extrinsic motivation could have in the process of self-regulation. Besides those theories, we present the Prisoner's Dilemma game, and its variation, the Iterated Prisoner's Dilemma game, that was deployed to simulate the interaction and the dilemma that is experienced during a self-control task. Finally, we present the reinforcing learning algorithm of Q-Learning that is deployed by our model. Chapter 3 is focused on the design and the implementation of the self-control model that consist of two Q-learning agents that represent the higher and lower parts of the brain. In addition, the methodology that is used for the simulation of each self-control theory is described in more detail. In Chapter 4 we observe the results of the simulations and clarify the influence on self-control that was found in each speculation. Finally, chapter 5 is an overview, where we synopsized the correlation between our findings and the existing information on each inspected theory.

Chapter 2

Epistemological Background

2.1 Self-control ……. 5

2.1.1 The top-down model of self-control ……. 7

2.2 The Prisoner's Dilemma (PD) ……9

2.2.1 Iterated Prisoner's Dilemma (IPD) ……12

2.3 Self-control Strength model ……13

2.3.1 Muscle Recovery theory ……14

2.4 Self-control Motivators ……14

2.5 Self-control Virtue ……..15

2.6 Reinforcement Learning ……..16

2.6.1 The Q-Learning Algorithm ……19

2.6.2 The ε-greedy policy ……20

2.1 Self-control

Morality is a set of rules that enable people to live together in harmony, and virtue involves internalizing those rules. If virtue depends on overcoming selfish or antisocial impulses for the sake of what is best for the group or collective, self-control can be said to be the master virtue (Beumaister,1997). In our daily life we are called many times to make decisions about the actions we are ought to take. Most of the time is easier for our brain to make subconscious-autopilot decisions that are coming mainly from our habits and our preferences. However, when it comes to changing our habits, is essential to become aware and conscious about the decisions we make in order to be able to stop our self from repeatedly following the same patterns of behaviors. Such decisions require from oneself to have Self Control and Self-discipline.

Self control behavior has received special attention and has been extensively researched by psychologists, behavioral economists, and neuroscientists in the search of revealing its mechanisms. The exact mechanisms have not been clearly defined, however many experimental findings could be pointing to the core of this behavior. In addition, throughout human evolution it is not obvious why perfect self-control would be difficult to evolve, after all self-control decisions are, ostensibly, just like another mental operation. One thing we know for sure however is that many socially problematic behaviors involve self-control failures, whereas the majority of positive virtues are based on high and effective self-control. In a study, it was found that young people high in self-control invest more time on doing their homework, have higher school attendance and focus and as a result get higher grades (Duckworth & Seligman, 2005). On the other hand, low self-control is connected to a variety of difficulties, including academic underachievement, an unhealthy lifestyle, procrastination and problems with the law (Moffitt et al., 2011). Self-control problems arise because human nature is not always rational as perceived in the context of economic theory which assumes rational utility agents (von Neumann and Morgenstern, 1947), otherwise the choice of the large delayed reward would have always been practiced.

The good news nevertheless is that human's skills are not predefined and the self-control behavior can for sure be developed and improved. For example, in a study (Muraven et al., 1999), a group of students was asked to regularly perform some easy self-control tasks for two weeks. These participants showed significant improvements on self-control compared with participants who did not practice self-control. Discovering our full potential in life will never be achieved if we cannot master the ability to control ourselves in the face of hardship and darkness. As the stoic philosopher Epictetus puts it, "No man is free who is not master of himself ".

2.1.1 Cognitive Neuroscience model of self-control

Self-control is defined as the dilemma between a smaller sooner reward (SS) and a large later reward (LL) (Rachlin, 2000), that is experienced as an internal conflict between the higher (prefrontal cortex) and lower (limbic system) parts of the brain . When we set a long term goal , we know from the beginning that we will be encountering such dilemmas where the sooner but smaller reward (SS) seems tempting and invigorating in the short term but that option is rarely aligned with the long term goal , that is the large but delayed reward (LL) . Figure 2.1 illustrates the concept of the delay of gratification. When faced with the dilemma of choosing the SS and the LL reward, one has to exert self-control and not surrender to the temptation at time t2 where the value of immediate reward becomes greater than the value of future reward. When we set a long-term goal, in the beginning (t1) the future reward (LL) has the greatest value since is the only option aligned with the goal. However, as time passes and the temptation is getting closer, the SS value is increasing and the preferences reverse.

This reversal of preference is a result of the discount mechanism the brain uses in order to determine value of any reward at the present moment. The longer the distance in time of receiving a reward, the more discounted is value appears and vice versa. That's the reason why the longer we managed to stick in the right track the easier it is to proceed until we complete the long-term goal. Let us consider a simple example of a student who faces the following dilemma. The student got an invitation for a party that takes place tonight, but he also has a quiz the next morning. Thus, he has to choose between the SS reward which is to accept the invitation, go to the party and have fun, and the LL reward which is to stay at home, focus on studying, perform well on the quiz the next day and ultimately have good grades at the end of the semester. In the previous scenario if the quiz was happening next week instead of next morning, the level of conflict that the student would experience is much smaller and the choice of accepting the invitation for the party seems more valid.

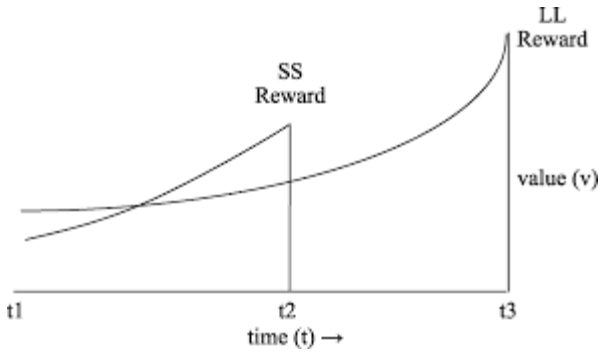

Figure 2.1: *Delay of gratification (based upon Rachlin, 1995). One has a choice between the SS reward at time t2 and the LL reward at time t3. The SS reward at time t2, its greater than the discounted LL reward. Achieving self-control means to wait for the bigger reward at time t3 and not succumb to the immediate smaller reward (t2).*

Cognitive neuroscience suggests a model of self-control (Figure 2.2) which depicts the internal process that an individual experiences, as our student in the previous example. In this model, the state of the environment is perceived from the higher center of the brain (prefrontal cortex) which is responsible for rational thinking, planning and control, and interacts with signals from the lower brain (limbic system), which is associated with our animal instinct and motivations, and is responsible for selecting an action. Finally, this internal conflict results in an action, which is rewarded or punished by stimuli from the external environment (Rachlin, 2000).

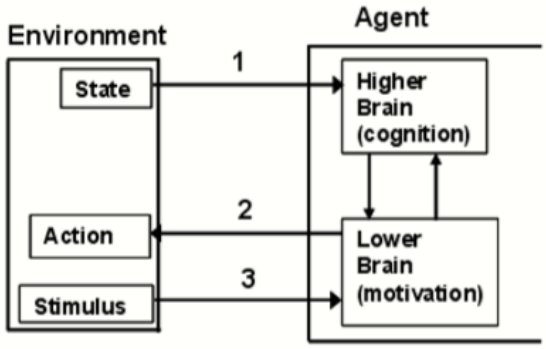

Figure 2.2: *A model of self-control behavior based upon Rachlin (2000). The higher brain perceives the state of the environment (Arrow 1). Through the interaction of the higher and lower brain an action is generated (Arrow 2). Stimulus reinforces the behavior by giving a reward or a punishment (Arrow 3).*

The activation of the prefrontal cortex while anticipating long-term rewards and the activation of the limbic system with immediate rewards, are shown in brain-imaging studies (McClure et al., 2004) and thus, they confirm that an internal conflict takes place and the need of exercising self-control. Furthermore, neuroscientists showed that there is increased activity in the prefrontal cortex (PFC) whenever one exerts self-control over temptation (Hare et al., 2009) .To simplify things ,we can imagine the two parts of the brain as two systems where the one is patient and knows that the longer it waits the better it is and the other is impatient and is driven by desire of receiving a fast reward (Kahneman, 2011).

2.2 Prisoners Dilemma

In order to model the internal conflict that two parts of the brain experience through their interaction we will be using a general sum game the Prisoner's Dilemma (PD) game. In the PD game the two agents have two options, one is to cooperate for a common good outcome and the second option is to defect for the service of personal interest. The nature of this game allows us to successfully describe the dilemma one could experience when faced with a self-control task.

The Prisoner's Dilemma game is described in the following scenario. A crime is committed and the police arrest two suspects - Merry and Louis. The police, however, do not have enough evidence to charge the two suspects so they decide to interrogate the two suspects in two different rooms. Both suspects are then given two options. The first is to remain silence, that is to cooperate with the other suspect, and the second option is to confess against the other suspect, that is to defect. If she decides to confess (defect), and he decides not to confess, then Merry will be rewarded with her freedom because she helped the police, and Louis will be sentence to 20 years in prison. The reverse scenario is also valid. In case that both of the suspects decide to defect, they will be sentence to 10 years in prison. The last scenario is when both Merry and Louis decide to remain silent and not confess (cooperate). In that scenario the police will not collect any further information that would reinforce the charges and the criminals will be sentence for only 2 years. Since both of them are in different rooms, they cannot know what their partner will choose to do. When both decide to act considering only their individual interest, they will both choose to confess against their partner. This strategic reverts them from risking to get the highest sentence (20 years)

but also gives them a chance to be set free if their partner decides not to confess. However since both will follow the same strategy of defecting, this will cause both of them to be sentenced to 10 years in prison. The dilemma arises from the fact that that there is better outcome than mutual defection and that is, the mutual cooperation. The strategy that is chosen by the players and from which nobody wants to deviate, is called a Nash equilibrium (Nash, 1950). According to Nash the best results comes from everyone in the group doing what's best for themselves and the group. The following figure shows all the possible scenarios of the PD.

		Louis: Cooperate	Louis: Defect
Merry	Cooperate	2, 2	20, 0
	Defect	0, 20	10, 10

Figure 2.3: *The number of years in prison for Merry and Louis according to the Prisoner's Dilemma game. If one of them defects and the other remains silent, the one who defected is set free, while his/her partner is sentenced to 20 years in prison. Merry and Louis get 2 years in prison if they cooperate and do not confess and 10 years if they both defect and confess .*

In our model the higher and lowers parts of the brain could be thought as two agents participating in a Prisoner's Dilemma game with each having to choose to either cooperate or defect. In that way the internal conflict that the parts of the brain experience during a self-control task will be represented through the PD rewarding matrix . The four possible states are extracted by the two possible actions:

- CC states is the result of mutual cooperation.
- CD or DC states are the result of one of the agents cooperating and the other defecting.
- DD state is a result of mutual defection.

Since CC is the best outcome for both agents, CC states are the goal states that represent self-control behavior. Below are the choices of the student who experiences a value conflict (have fun or have good grades), and how it resembles with the PD game (Christodoulou et al., 2010):

a. Go to the party, which corresponds to cooperation from the brain's lower part.
b. Stay home and study, which corresponds to cooperation from the brain's higher part.
c. Go to the party for some time and return home to study, which corresponds to mutual cooperation (CC state).
d. Do nothing, which corresponds to defection from both sides (DD state).

Each outcome is accompanied with different rewards (payoff values) for the two players. The four possible rewards an agent could receive are the following:

- Temptation payoff (T) – The agent's reward for defecting while the other agent cooperates.
- Reward payoff (R) - The agent's reward for mutual cooperation
- Punishment payoff (P) – The agent's reward for mutual defection
- Sucker's payoff (S) - The agent's reward for cooperating while the other agent deflect

The basic rule that must be satisfied in a PD game defines the following inequality:

T > R > P > S (1)

Figure 2.4 shows the payoff matrix of the Prisoner's Dilemma game.

		Column Player	
		Cooperate	Defect
Row player	Cooperate	R, R	S, T
	Defect	T, S	P, P

Figure 2.4: *The payoff matrix of the Prisoner's Dilemma game. Each player either cooperates or defects. When both cooperate, they both receive a Reward payoff (R), but when they both defect, they receive a Punishment payoff (P). When one of the players cooperate and the other defects, they receive the Sucker's payoff and Temptation payoffs, respectively.*

2.2.1 The Iterated Prisoner's Dilemma

In our implementation we will be using the Iterated Prisoner's Dilemma (IPD) that is a game where the one-shot PD is played consecutively by the two agents. The repeated version suits the interpretation of intra-personal conflict more realistically as the two internal agents compete with each other for more than one time. Additionally, the duration of the game allows incorporating learning in the decision-making process and the agent's strategy can differs from before. The design of the game requires the following extra rule to be always satisfied throughout the game.

$$2R > T + S \quad (2)$$

This rule guarantees that CC states are indeed the best option for the organism since the players are not collectively better off by having each player alternate between "Cooperate" and "Defect". During each round of the IPD, the states of the two agents represent the decision that the person took to solve his/her dilemma. Our goal state is the CC state, since is the state that both agents cooperate and self-control is achieved. The implemented agents are rational, meaning that they pursue reward maximization during the game according to their own value systems that satisfy their individual goals, giving no interest on the well being of the other agent or the organism. Fortunately, the well being of the organism is in line with their individual well being in the long run. This is because the agents can achieve long term payoff maximization only through mutual cooperation which maximizes payoff for the whole system as well.

Other general-sum games than the PD game like the Battle of the Sexes game (BoS) or the Rubinstein's Bargaining Game (RBG) (Rubinstein, 1982), could be considered to simulate the interaction between the higher and the lower brain . However , it is proven empirically that the PD game is for now the optimum option in modeling self-control internal conflict — Also employed by Banfield (2006), Cleanthous (2010), and Christodoulou et al. (2010).

2.3 Self-control Strength Model

Self control was something that concerned the psychologists for many years and they tried to provide a theory that explains the self control failure. The idea that self-control resembles a muscle that its exertion depletes a limited resource makes specific predictions with respect to self-control failure (Muraven and Baumeister,2000). Muraven and Baumeister embraced the idea that self control should be viewed both as a muscle and as a skill and should be trained as such in order to be improved. They introduced the "self-control strength model". According to this model, the ability to exhibit self-control relies on a limited resource, or self-control strength, and all different self-control operations draw on that same resource.

In their previous study, Muraven et al. (1998) demonstrated that participants' performance was impaired in a self-control task that followed an initial one. In addition, the impairment was found even if the two tasks were completely different in context. The model's view of self-control resembles a muscle that its short-term ability decreases after exertion but at the same time repeated exercise strengthens it in the long run. In another study (Muraven et al., 1999), a group of students was asked to regularly perform some easy self-control tasks for two weeks. These participants showed significant improvements on self-control compared with participants who did not practice self-control. Finally, Dewitte, Bruyneel and Geyskens, (2009) found that depletion effects were reversed when the two subsequent self-control tasks involved similar response conflicts. In three studies, they showed that when adapted to a demanding situation, people's self-control performance decreased in situations when the subsequent self-control task involved a different response conflict (replicating the typical depletion effects) but improved when the subsequent self-control task involved a similar response conflict.

Muraven et al. (1999) did not specify the nature of the limited resource on which self control relies on. However, a series of experiments confirmed that willpower and self control are tied to glucose (Gailliot et al., 2007). The study showed that reduced blood glucose and poor glucose tolerance (reduced ability to transport glucose to the brain) are linked highly with lower performances in self-control tasks. In addition, it was confirmed that the effects of ego depletion (depleted energy resources) can be counteracted by giving people a dose of glucose.

2.3.1 Muscle Recovery

Recovery is the single most important part of any training or exercise program. Muscle recovery is an improvement in the performance of your skeletal muscles utilized during prolonged or intense exercise [1]. After a workout, the body replaces, or repairs damaged muscle fibers through a cellular procedure where it tempers muscle strands together to make new myofibrils or muscle protein. De-loading is another term for giving your muscles a rest [2]. However, it is not a complete rest, where you stop training for a time, but a reduction in the stress you place on your muscles. It's based on the concept of supercompensation.

When we train, we apply stress to your muscles by forcing them to work harder than they are accustomed to. This creates muscle fatigue and neurological or brain fatigue. During the rest or recovery phase between workouts, your muscles recover. The idea behind the de-load is to give your muscles a longer break where you still train but with less intensity. In response, your muscles enter a phase called supercompensation where they bounce back to a higher level of performance. For instance, let us take a person who works out each week in a high intensity training program could choose to de-load by choosing 1 week of each month where he will keep training but this time with a more relaxed and less intense training program. On a self-control task however, intensity is described by the intensity of the internal conflict that the individual experience. Assuming that the self-control muscle theory is valid, we can examine whether muscle recovery theories could be applied in our model in order to increase the self-control strength.

2.4 Self-control Motivators

Psychology distinguishes between two broad classes of motivation to perform an activity: intrinsic motivation and extrinsic motivation. A person is intrinsically motivated if he performs an activity for no apparent reward except the activity itself (Berlyne, 1966). Extrinsic motivation, on the other hand, refers to the performance of an activity because it leads to external rewards (e.g., status,

[1].http://www.yourwebdoc.com/musclerecovery.php
[2] https://www.muscleandstrength.com/expert-guides/strength

approval, or passing grades). The ability to show self-control is deeply linked with both classes of motivation.

For instance, let us take a smoker who tries to quit smoking for a certain period. The intrinsic strategy that he could follow is to study the benefits and overall the reasons why quitting smoking is a good thing for him. In that way, he will more internally motivated to show resistance in the face of temptations. On the other hand, the extrinsic strategy would be to have an external motivator that offers the smoker an external reward (e.g., money) in case he manages to persist in abstaining from smoking. Elimination of the depletion effect has been found when participants are offered the monetary incentive to perform better on the persistence task (Baumeister, 2006), informed that persistence would improve their performance on a target task, or believe that persistence is warranted because the issue requiring persistence is of substantial importance (Muraven and Slessareva 2003).

2.5 Self-control Virtue

Self-regulation can be analyzed into three main ingredients: standards, monitoring, and operations that alter the self (Baumeister & Exline, 2001). Failures or problems with any of these ingredients can result in the breakdown of self-control (e.g., Baumeister & Heatherton, 1994; Baumeister, Heatherton, & Tice, 1996). Virtue thus depends on three factors. The first is having clear standards. Much of Western literature has revolved around portraying people facing moral dilemmas, because such dilemmas reflect the difficulty of doing what is right when moral standards are lacking or, more commonly, in conflict. In our case, when people experience a dilemma between choosing a sooner tempting SS reward or choosing to wait for the later but more rewarding future reward (LL), the standards of the individual set the amount of the internal conflict that exist.

Second, virtue depends on monitoring, which is a matter of keeping track of one's own behavior and comparing it to the relevant standards. Self-regulation is thus closely allied with self-awareness (Carver & Scheier, 1981). Circumstances that cause people to stop monitoring their behavior are likely to reduce virtuous behavior. An example of these circumstances could be the presence of negative emotions. Studies showed that negative emotions such as anger, anxiety, fear and sadness often reduce self-control (Cyders & Smith, 2008; Heatherton, 2011; Schmeichel &

Tang, 2015) .By the same token, alcohol reduces self-awareness (Hull, 1981), and alcohol is well known to be implicated in a broad range of nonvirtuous behavior ranging from interpersonal violence to sexual misdeeds (Baumeister, 1997; Baumeister, Heatherton, & Tice, 1994).Therefore the ability of monitoring the self and his relevant standards is an important factor on sustaining the virtuous behavior , that is self control.

Third, virtue depends on the self's capacity to alter its own behavior so as to conform to standards. Violent impulses must be restrained, promises must be kept even despite disinclinations, temptations must be resisted, and so forth. Even if the self has clear standards of virtue and understands how they apply to its current situation, behavior may fall short of virtue if the self is unable to make itself behave according to them. In other words, self-control and self-regulation relies on the capability to recorrect oneself and align one's behavior with the standards, relevant to the desired outcome.

2.6 Reinforcement Learning

Reinforcement Learning (RL) is one of the three categories of learning that exist (Supervised, Unsupervised, Reinforcement Learning). It constitutes of a class of algorithms that can be used in order to train an agent to maximize his accumulated reward, through the interaction with an environment (dynamic or static) where the only source of feedback is a reinforcing signal. The reinforcing signal could be either a reward or a penalty. Since the goal of the agent is to maximize his accumulated reward, each reinforcement signal could either impact positively or negatively towards following a certain strategic. This concept underlies all human learning theories and is considered to be the most "natural" way of learning. This learning is also a trial-and-error learning. For example, an infant who tries to touch the fire for the first time will burn and the reinforcement signal(pain) that he will receive would motivate him to avoid doing the same action again.

Reinforcement learning is also applied when we as human behave in society, throughout the interaction with other people we change our behaviors and actions in order to adapt to the environment situation , to maximize our chances of achieving are goals and overall to avoid receiving any form of a penalty .Reinforcement learning differs from the other two categories of

learning (Supervised and Unsupervised) , since there is no learning towards a predefined desired output and is the most efficient learning method when dealing with a dynamic problem – environment . In RL, the agent does not know a priori an optimal policy and is not told which actions to perform. For this reason, the agent has to explore the environment and learn which strategic of actions is more rewarding. During the course of the interaction with the environment, before choosing an action the agent must decide whether he will use the knowledge he already obtained through learning (exploitation) in order to choose the best next action or whether he will explore and try an action he never tried before. This is the exploration-exploitation trade-off; in order to learn it has to explore, but in order to perform well it needs to exploit what it already knows (Woergoetter & Porr, 2008). Generally speaking, the best approach most of the times is to be optimistic over uncertainty in the beginning of learning, that is to explore more , and then as learning takes places to tend to be making good use the obtain knowledge.

The formal framework of Markov decision processes (MDP) is used to define the interaction between a learning agent and its environment in terms of states, actions and rewards. Sequential decision-making problems where actions influence subsequent situations are formalized by deploying MDPs. In the following figure we can see an example of the interaction of an agent with the environment. The interaction is a cyclic relation between the agent's action and the reinforcement signal that the action produce. To be more precise, the agent at the start of each round chooses an action α, by considering both the environment's current state and his obtain knowledge. According to the action that is then taken, the environment would change state and a reinforcement signal will be send back to the agent. Since the agent's objective is to maximize the accumulated reward over time, it forms a policy by which the agent selects actions as a function of states. The goal is then to find the optimal policy where a value function links each state-action pair with the largest expected return.

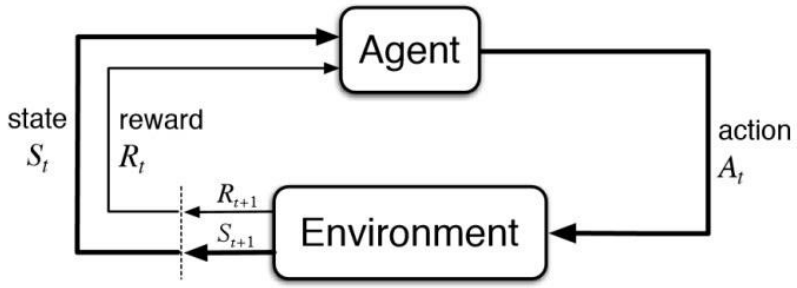

Figure 2.5: *The reinforcement-learning model. At each step t, the agent selects an action A_t according to the state S_t and the reward R_t. The action will then have an impact on the environment from which the next state S_{t+1} and reward R_{t+1} will be induced.*

In our case we deploy temporal-difference (TD) learning (Sutton, 1988) which combines the Dynamic Programming (DP) and Monte Carlo methods. TD methods are model-free and learn from episodes of experience like Monte Carlo methods and they update estimates without waiting for the final outcome by using bootstrapping methods similarly to DP.
The update rule of TD is equation (1):

$$V(S_t) \leftarrow V(S_t) + \eta \left[G_t - V(S_t) \right] \qquad (1)$$

V(St) is the expected reward - value when we are in the state *St* , n is the learning rate and *Gt* is the actual returned reward . The expected reward - value for each state is updated considering both the previous estimation and the actual reward. In our model we will be using the Q-Learning algorithm which belongs to the TD method.

2.6.1 The Q-Learning Algorithm

The Q-Learning Algorithm is belonging to the TD methods and is one of the most common methods used for estimating the expected reward of a state-action pair, that is the Q-value function. The basic idea of Q-Learning method is always following the action that is so far more rewarding according to our current knowledge and in therefore minimize the chance of a good move, followed by a bad move. Its update rule is the following equation (2):

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \eta \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right] \qquad (2)$$

The Q-learning algorithm follows the same logic we the basic TD methods. Each time when the agent is in a state St and takes an action At then the expected Q-value of the state-action pair, that is the Q(St,At) , is updated according to this rule . At time t+1 after the action is taken the agent receives the reward R_{t+1} and this reward combined with the discounted estimate optimal future reward are consider in the new Q-value . The η term is the learning rate, which defines how much change will be consider in the new Q-value. The future optimal reward is the $\max_a Q(S_{t+1}, a)$, which is the maximum expected Q-value considering the next state St+1. This future reward is discounted by the parameter $\gamma \in [0, 1]$. The discount rate γ is an important parameter since is the parameter that defines the whether the agent is myopic and focus more on the immediate or long-sighted and focus more on the future rewards. Values of the γ parameter closer to 1 indicate that the agent focus on the future (or LL) rewards, whereas values closer to 0 indicate that the agent is more myopic and focus on the immediate (or SS) rewards.

Q-Learning is an off-policy TD control algorithm, because no matter the agent's starting policy the Q-Learning update rule will always be predicting that the best possible action α will a be taken in each next state. In that way the algorithm guarantees that the optimal policy and the optimal value function are found. Other TD methods could be considered in the self-control model, however Georgiou (2015) model self-control using both Q-learning algorithm and SARSA method and showed that the agents learnt to exercise self-control more easily with the Q-Learning algorithm in comparison with the SARSA algorithm.

2.6.2 The ε-greedy policy

Throughout the game the agents must have a policy for deciding between exploitation – choosing an action based on agent's knowledge, or exploration – randomly choosing an action. Choosing only to explore is not a valid option since we will never be able to find converge to an optimal strategic. On the other hand, choosing only to exploit is not a valid option either since this will mean that the agent will follow the same actions and no further learning will take place. For that reason, we need a good policy that can solve the exploration-exploitation problem. The policy that is going to be used in order to resolve this dilemma is the ε – greedy policy. The e-greedy policy defines that in each round the agent will choose to explore a new random action with a probability ε, or will choose to exploit and use the already accumulated knowledge with a probability 1-ε . For example, with ε-value= 0.2 , the agent will choose with probability 0.2 to explore and with probability 0.8 to exploit.

$$A_t = \begin{cases} A_t^* & \text{with probability } 1-\varepsilon \\ \text{random action} & \text{with probability } \varepsilon \end{cases}$$

Figure 2.6: *The e-greedy policy of the agent. Exploit (At*) action with probability 1-ε , Exploration Action with probability ε.*

Chapter 3

Design and Implementation

3.1 Introduction ……21
3.2 Simulating Self-control Strength Model ……22
3.2.1 Muscle Recovery ……23
3.2.2. Subsequent Self-control Tasks ……24
3.3 Simulating Extrinsic Motivation ……25
3.4 Simulating Self-control as a Virtue..……26
3.5 The Q-Learning agents ……..26
3.6 The Q-Learning model ……27

3.1 Introduction

In this chapter we analyse and explain the methodology that is being used in order to simulate the findings align with the self-control muscle and virtue theory. We discover how the effects of ego depletion (depleted self-control resources) appear in our model. Furthermore, we determine the method that is used to simulate the self-control muscle recovery periods and self-control task switching. We also examine the method on which the presence of an external motivator could be simulated as well as the design aspects that are used on modeling self-control as a virtue. Finally, we display our baseline self-control program along with the methods that are used for each different simulation. The development of Q-Learning Agents is entirely based on the work of Georgiou (2015), who created the class *Player* and the main class is an extension of the *Main* class used by Nikodemou (2020). The programming language that is used is Java.

3.2 Simulating Self-control Strength Model

In the following subsections we discover how some theories associated with self-control "strength model" are simulated. Before we proceed however, it is important to clarify the nature of the computational model that is used. We deployed a reinforcement learning model that consists of two Q-learning subagents that interact in the IPD game with the goal of learning to cooperate , and thus achieve self-control .Banfield (2006) and Cleanthous (2010) , modelled self-control behaviour using neural networks and it was proven that irrespective of whether we use a biological realistic spiking neural networks or a simpler model , self-control is successfully modelled and the results remain the same (Christodoulou et al., 2010) .

According to the self-control "strength model, self-control resembles a muscle and the ability to exhibit self-control relies on a limited resource or self-control strength. Self-control strength is limited, in the sense that a person has finite capacity for self-control and can override only a finite number of urges at the same time (Muraven and Baumeister 2000). Ego depletion refers to the reduction of such resources as a result of the depletion of self-control muscle during a task or a sequence of tasks. Apart from the intensity of the internal conflict that the brain experiences during a self-control task, the duration of the task has a major influence also on whether the desired behavior is achieved. The duration of a self-control task is described by the total of rounds of IPD game, in which the two Q-learning agents participate. By increasing the total number of rounds in the IPD game, we expect to notice a reduction on the CC states that represent self-control as a result of the further depletion of the brain. Moreover, we expect to observe a constant decrease on CC states when the system reaches the depletion period during the game. (Muraven, M. and Baumeister, R. F, 2000). The muscle depletion is explained in more detail in section 2.3.
Moving on to the next subsections, we examine how these fatigue effects could be limited in our model by incorporating the concept of recovery and extrinsic motivation in our simulations.

3.2.1 Muscle Recovery

The strength model entails that the available stock of resources is depleted by exertion and must be replenished before the full measure is available again (Muraven and Baumeister, 2000). It thus resembles a muscle that becomes fatigued by exertion and becomes less able to function. Simulating recovery periods in which the agents retain their strength resources cannot be achieved by simply pausing the IPD game, since the concept of recovery is not implemented in the first place. For that reason, we need to follow a different approach where the agents continue to participate on the IPD game while experiencing a recovery state. Therefore, recovery states are simulated as states in which the agents continue to interact but experience a decreased internal conflict .

Cleanthous (2010) indicated that the increment of the difference between the Temptation (T) payoff value and Sucker's (S) payoff value (T-S) , increases the internal conflict that the system perceives , whereas the decrement of T-S , decreases it . We used that knowledge to implement the reduction in conflict accordingly. Finally, muscle recovery is simulated by changing periodically the payoff matrix during the IPD game from the initial baseline matrix to the recovery state matrix for a certain recovery period (certain amount of rounds). By incorporating muscle recovery in the model, we expect to notice an increase in CC states in comparison with the default execution where only a constant payoff matrix is used. Figure 3.1 shows the two states of the payoff matrix that exist when training is combined with recovery. The conflict intensity of the payoff matrix is modified when we change the positive values X and Y.

		Agent 2	
		Cooperate	Defect
Agent 1	Cooperate	R, R	S, T
	Defect	T, S	P, P

		Agent 2	
		Cooperate	Defect
Agent 1	Cooperate	R, R	S+X, T-Y
	Defect	T-Y, S+X	P, P

Figure 3.1: *The payoff matrix of the Prisoner's Dilemma game. (Left) The default payoff matrix that the agents have when they are not recovering. (Right) The recovery payoff matrix that the agents have while recovering. Note that X and Y are positive values and as they increase the intensity of the internal conflict of the payoff matrix decreases.*

3.2.2 Subsequent Self-control Tasks

In our model a self-control task is described by two elements, the payoff value matrix which describes the difficulty of the task, and the duration in rounds of IPD game that the task lasts.
We simulate accordingly the self-control task switching by changing the payoff matrix for the subsequent task. Dewitte , Bruyneel and Geyskens (2009) showed through their studies that self-control performance differs when a subsequent task involved similar or different response conflict. To simulate a subsequent task that has a similar internal conflict to the initial one, we add a random constant value to each payoff value of the initial matrix (T,R,P,S) . By doing that the two rules of IPD game are unaffected since the correlation of each payoff value remains still and at the same time the agents experience a completely new task. On the other hand, when it comes to simulating a subsequent task with a different internal conflict we change only the difference between the Temptation (T) payoff and the Sucker's (S) payoff value . An extra rule guarantees that the random updated T and S values do not violate the rules of the IPD game (section 2.2).

In the first case where the subsequent tasks follow a similar internal conflict with the initial one, we expect to notice an increase in self-control (CC states) . Alternatively, when the following tasks hold a different internal conflict we expect to notice a decrease in self-control. On the occasion that our assumptions are correct, the results would be aligned with the experimental findings on the related performed studies (Dewitte, Bruyneel and Geyskens, 2009)

3.3 Simulating Extrinsic Motivation

Extrinsic motivation refers to the performance of an activity that is driven by an external reward (or an external factor) independent of the reward of the activity itself (Berlyne, 1966). Such external rewards could be status, approval, money and more. The first simulation related to extrinsic motivation has to do with an external monetary reward that appears in the IPD game just as the effects of ego depletion appear that cause a constant decrease in CC states. To simulate this external reward, we change the discount rate of the higher brain agent as soon as the effects of depletion appear during the IPD game. By doing that the upper brain agent will give greater value to the large later (LL) reward that's the desired reward for the system. Remember, the lower brain agent is driven by default to the short-term reward as it represents the limbic system that is associated with our animal instincts and motivations.

The second simulation is associated with an external authority that changes the nature of the game by forcing the system to cooperate more. This is achieved by forcing the agents to choose periodically cooperation for some rounds during the IPD game. For both simulations, we expect to notice an increase on CC states in comparison with the baseline execution since the external motive drives the organism towards the behavior of self-control.

3.4 Simulating Self-control as a Virtue

Self-control virtue theory suggests that self-regulation can be analyzed into three main ingredients: standards, monitoring, and operations that alter the self (Baumeister & Exline ,2001). Before, viewing how the self-control virtue theory is incorporated in our model is important to understand the way in which the presence of negative emotions is implemented in our model. Nikodemou (2020) was the first to implement the concept of emotions and she proven that the presence of negative emotions is successfully simulated by periodically increasing T (Temptation) and/or decreasing R (Reward) payoff value during the IPD game. Both strategies guarantee a decrease on self-control (CC states).

In our model we will be combining the presence of negative emotions with the self-control virtue theory. The three ingredients previously mentioned that describe self-control as a virtue can be distinguished into two elements. The first element is the relevant standards that are described by the initial payoff matrix before the effects of the presence of negative emotions are applied. Everyone can experience the same self-control task differently depending on his/her relative standards. The second element refers to the ability of the system to alter its own behavior so as to conform to relevant standards. This is simulated by resetting the payoff matrix back to it is initial standard values, while the system continues to experience the presence of negative emotions. Through this simulation, we expect to notice an increase on self-control compared to the executions where only the effects of the presence of negative emotions were applied. Such outcome would reinforce the beliefs associated with the self-regulation virtue theory.

3.5 The Q-Learning agents

The Q-learning subagents are objects of the *Player* class. The model consists of two agents that represent the lower and higher parts of the brain that interact through the IPD game. The state of the object is described by the following fields: the learning rate (η), the epsilon value which defines the e-greedy policy, the discount factor (γ), the Q-table that the agent keeps for the estimated value for every state-action pair , the payoff matrix and the current action that keeps the agent's chosen action. It's important to note that the discount rate is the parameter that is set differently to

distinguish the one agent from the other (see Section 2.4.1). The Player's Q-table and the payoff matrix is initialized through the constructor. The Q-table consists of 4 rows with 2 columns each. This is due to the fact that there are 4 possible states (CC,CD,DC,DD) and 2 actions that the agent can choose to take , that is to Cooperate (C) or Defect (D) . Setters and getter methods for the fields of the object are also implemented.

The exploration-exploitation problem was solved using the e-greedy policy (section 2.4.2) and is implemented in the method *choose_action*. Lastly, the core method of the class *Player* is *update_Q* . Through this method learning takes places by applying the Q-learning algorithm (section 2.4.1) to update the values of the Q-table .

3.6 The Q-Learning model

The Q-Learning model is created through the *MainQ* class. More specifically in the *MainQ* class we decide which function is to be called depending on the theory (e.g. muscle recovery, extrinsic motivation) that we want to investigate. We will analyze each method in more detail but is essential to underline that each function has same structure before and after their main body. The code of the program along with each function is at the appendix A. Before the main body of any of these functions by default we set the model to run for 15 trials of 1000 episodes each. In the beginning we initialize 3 arrays : *states_results* to hold the states that the agents reach during the game , *payoff_results* that have the rewards each agent earns from the payoff matrix and the *overall_payoff* array with the overall payoff of the agents. When the learning is done, the results stored in these arrays are uploaded in text files. The two agents are then initialized through the constructor of the class *Player* (section 3.4) . Proceeding we set the payoff matrix of both agents with the values T,R,P,S accordingly and also the epsilon value of the e-greedy exploration . To distinguish the two agents we set the discount factor (γ) to 0.1 for the agent that represents the lower part of the brain (the limbic system) and to 0.9 for the agent that represents the upper part of the brain (prefrontal cortex) . In this manner the lower brain agent becomes myopic and is driven towards the short term reward (SS) , whereas the upper brain agent becomes long-sighted and is driven towards the long term reward (LL) . In every round of a trial both agents select an action based on their current state, which is then used to update their Q-tables.

As we mentioned previously, when the learning takes place we call a different method depending on the theory we are analyzing.

The functions related to self-control "strength model" are the following :

- *Round_depletion* : This function takes only 1 parameter as input . The parameter is an integer and it represents the total of rounds of IPD game that the agents will interact in every trial . This method is used explore the fatigue effects that appear when the duration of the same task changes.
- *Muscle_Recovery* : The function takes 2 integers as parameters . The first parameter is the variable *recover* that represents the recovery period in rounds of the IPD game and the second argument is the variable *maxDays* that indicates the maximum rounds before the agents enter a recovery state. When in recovery the agents experience a lower conflict payoff matrix than the standard payoff matrix that is used otherwise (section 3.2.1).
- *Task_Switching* : This function contains 2 variables as arguments . The first argument is the variable *Similar* that is a 28oolean value indicating whether the system will be switching to a similar conflict payoff matrix for the next task (when is true) or we will be switching to a different conflict payoff matrix (when is false). The two scenarios are explained in more detail in section 3.2.1. The second parameter is the variable *rounds* which is the interval in rounds before the system shifts to a new task .

The next two functions are mainly related to the theory of extrinsic motivation.

- *External_Monetary_Reward* : This function has only 1 parameter . The parameter is an integer variable named *discount_interval* that defines the round in which the discount rate of the lower brain agents changes as a result of the external reward that showed up. The discount rate is then decreased by 0.3.
- *External_Motivator* : The following function takes only 1 argument as input . The argument is an integer variable named *forced_rounds_interval* that defines how often the agents are forced to cooperate as a result of the external authority forcing them to do so.

The final function is related to the self-control virtue theory as well as the concept of the presence of negative emotions used by Nikodemou (2020).

- *Clear_Standards* : The next function takes 4 arguments as parameters . The first argument is the variable *payoff_interval* that defines the number of rounds between each change of R or T payoff value. The ratio that these payoff values change is defined by the third and fourth argument respectively (ratio_R , ratio_T). Finally, the second argument is called *standards_interval* and it determines the number of rounds that should pass each time before we reset the payoff matrix back the initial one.

Despite the function that is used, in all cases we save the states that the agents reached and their returns. The findings are then saved to the *payoffs.txt* and *states.txt* text files . In the file *payoffs.txt* we save in 3 columns the accumulated payoff of the lower part of the brain, the higher part, and their sum, respectively. In the file *states.txt* each of the 4 columns are the average percentage of the frequency of the CC, CD, DC, and DD states, respectively. Based on these two files we produce the figures that describe the results of Chapter 4.

The three types of figures we will be using for our analysis in the next chapter are the following:

1. A line plot with 4 traces, with each trace representing one of possible outcome states. With this plot we can observe how the preference of the system changes throughout the rounds as well as the behavior that the system shows when any of the changes previously described are applied.
2. A bar plot which shows the overall average outcomes, that is, the percentage of the appearance of each state. The information that we obtain from this chart is only the difference between the outcomes and which state dominated at the end.
3. A line plot which shows the overall accumulated payoff (3rd column in the *payoffs.txt*) achieved by both agents during the game. The achieved outcome is compared to the theoretical best outcome.

Chapter 4

Results and Discussion

4.1 Introduction ……31

4.1.1 Constant payoff matrix ……31

4.2 Simulating Self-control Strength model ……33

4.2.1Ego depletion ……33

4.2.1.1 Increasing the Total number of rounds of IPD ……34

4.2.1.2 Increasing the Intensity of Internal conflict ……..36

4.2.2 Simulating Muscle Recovery ……37

4.2.3 Simulating Task switching ……41

4.2.3.1 Switching to Similar Intensity conflict tasks ……41

4.2.3.2 Switching to Different Intensity conflict tasks ……..43

4.2.4 Summary and discussion on Self-control Strength model ……45

4.3 Extrinsic Motivation ……46

4.3.1Simulating External monetary reward ……..47

4.3.2 Simulating External motivator ……49

4.3.3 Summary and discussion on Extrinsic motivation ……52

4.4 Simulating Self-control as a virtue ……52

4.4.1Presence of Negative emotions ……53

4.4.2 Simulating Clear standards ……..55

4.4.2.1Decreasing the Reward payoff ……56

4.4.2.2 Increasing the Temptation payoff ……58

4.4.2.3 Increasing Temptation and Decreasing Reward payoff ……60

4.4.3 Summary and discussion on self-control virtue theory ……62

4.1 Introduction

In this chapter we will be reviewing the results from each simulation related to the self-control strength model, the extrinsic motivation and self-control virtue theory as the two Q-learning agents participate in the IPD game. Each concept is was previously analyzed in more detail in Chapter 3 (sections 3.2 and 3.3).

4.1.1 Constant payoff matrix

Before moving forward to the results from each implemented method is essential to underline our baseline model. The basic model of self-control was first described in the thesis of Georgiou (2015) where the learning rate was set to 0.1 and the epsilon value to 0.1. The basic model has also a constant payoff matrix that holds the values T= 5 ,R= 4, P= -2 , S = -3 . The program runs 15 trials of the game with each trial lasting 1000 rounds of the IPD game. We will be using the same payoff matrix as the initial one for almost every following experiment. According to Cleanthous (2010), these chosen values for the initial payoff matrix represent a self-control task with an internal conflict of moderate intensity. The baseline results are presented in Figures 4.1 a-c. Figure 4.1a shows the average states achieved during the rounds of the IPD game and Figure 4.1b represent the overall outcome of the appearance of each state after 1000 rounds. Lastly, Figure 41c shows the overall performance of the Q-learning agents.

The accumulated payoff of a given round is calculated by adding together the payoff each agent received during a round according to the payoff matrix. For example, if the outcome for a given round was CC, then both agents will receive the reinforcement signal R (4) and the total accumulated payoff that will be added is R+R=4+4=8. The overall performance (Figure 4.1c) comes from the addition of the accumulated payoffs of each agent at each round , and then we calculate the average accumulated payoff when we divide with the number of the trials. The theoretical best performance is achieved when the system always chooses mutual cooperation (CC states) and in that case the maximum accumulated reward is 8000 (8 *1000) . In all cases, we compare the obtained performance with the theoretical best performance (dot-dashed line).

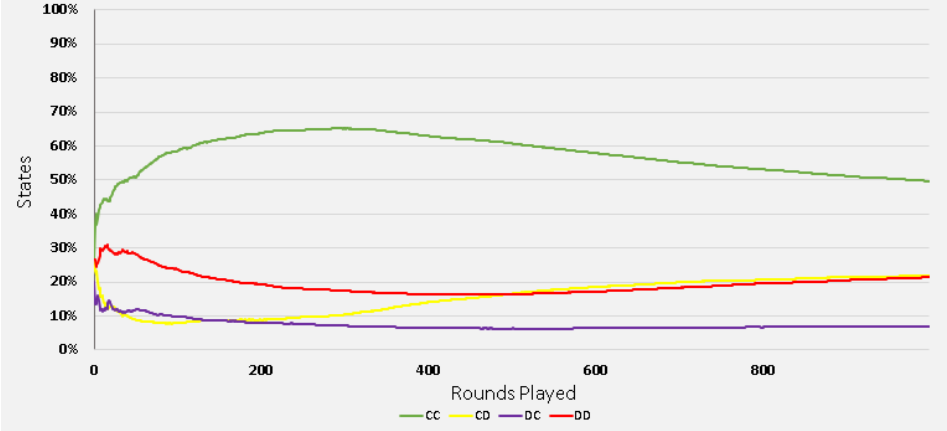

Figure 4.1a: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Constant Payoff Matrix (T=5, R=4 , P=-2 , S=-3) .*

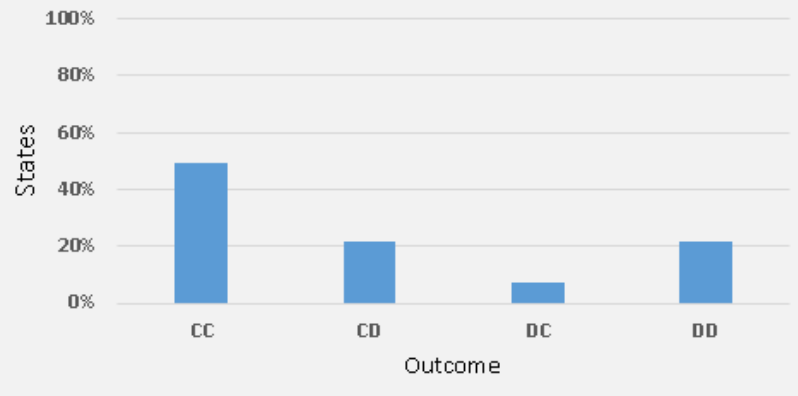

Figure 4.1b: *Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game. The constant payoff matrix is used: T=5, R=4, P=-2, S=-3.*

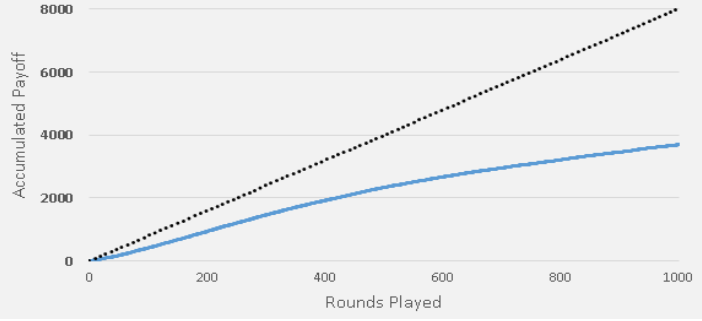

Figure 4.1c: *Overall performance of the Q-learning agents during the 1000 rounds of the IPD game (blue line). The theoretically best performance is shown for comparison (dot-dashed line)*

4.2 Simulating Self-control Strength model

In this section, we are focused on the results of the simulations related to the theory of self-control "strength model ". According to this model, self-control resembles a muscle and has limited resource or self-control strength, in the sense that a person has finite capacity for self-control. Thus, success or failure of self-control depends on the person's level of self-control strength. For example, people who have more strength should be more likely to reach a self-control goal, such as losing 10 pounds, than people who are lower in strength. However, although in the short–term self-control resources are depleted, repeated exercise strengthens the available resource in the long-run (Muraven et al., 1999).

The first simulation related to the previous concept aims on investigating the nature of self-control resources as well as the behaviour that our model exhibits when the resource is depleted. Secondly, we will be viewing the results achieved when we deploy the concept of muscle recovery in our model (section 3.2.1) . Finally, the last simulation linked to the concept of self-control strength model is focused on the performance in self-control that is achieved when the system shifts from the initial task (the initial payoff matrix) to subsequent tasks. In every simulation, the results could vary when different combinations of arguments are given. However, in all cases we will be using the same initial payoff matrix, learning rates and epsilon value as we used for our baseline results (section 4.1.1).

4.2.1 Ego depletion

A self-control task is described in our model by two factors. The first is the duration of the task, that is defined by the total number of rounds that the IPD game lasts in each trial. The second factor is the internal conflict of the task, that is represented by the values of the constant payoff matrix that is used in the course of the game. In the experiments that follow, we are using the method *Round_depletion* to discover the implications on self-control when we change these two factors.

4.2.1.1 Increasing the Total number of rounds of IPD

We used our baseline constant payoff matrix (section 4.1.1) to observe the influence on self-control when the total number of rounds is increasing. At the first attempt we set the total number of rounds to 500.That is 500 rounds less that our baseline. The obtained CC states reached the 60% of all the outcomes, which is around 10 percentage more than our baseline results (figures 4.2a,4.2c). In the second experiment we increased the total number of rounds of the IPD game by 500 compared to the baseline. When 1500 rounds were used in each trial, self-control behavior was not achieved since the achieved CC states reached only the 42% of all the outcomes. When we take a look at the 3 trace plot figures (4.1a ,4.2a ,4.2b), we observe that despite the three different results we obtain in the end of each experiment, the behavior that the system demonstrated throughout the rounds was similar. At first the system converges toward the desired behavior of self-control (CC states). Later, it reaches a peak on the CC states approximately at 250 rounds and right after that there is a constant decrease on self-control.

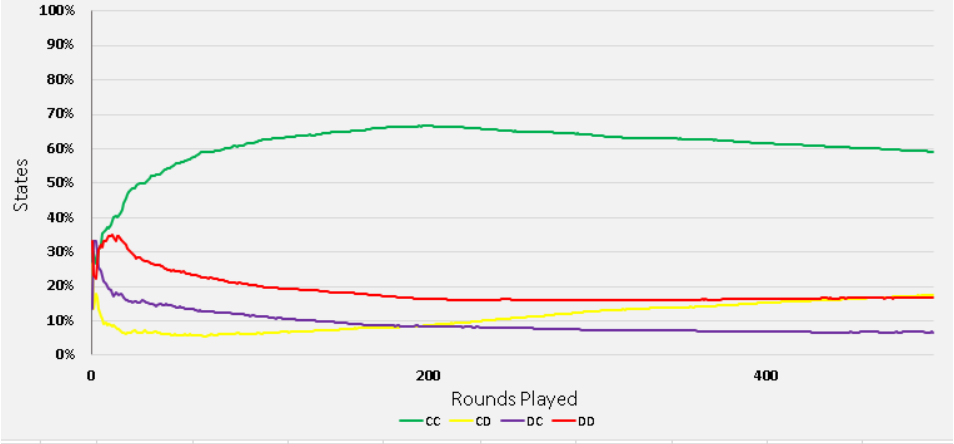

Figure 4.2 a: *Average of the outcomes CC, CD, DC and DD during 500 rounds of the Q-learning agents playing the IPD game. Constant Payoff Matrix (T=5, R=4 , P=-2 , S=-3) .Self-control states (CC) reach the maximum around 200-300 round then the system starts experiencing a constant decrease on CC states (Ego depletion period).*

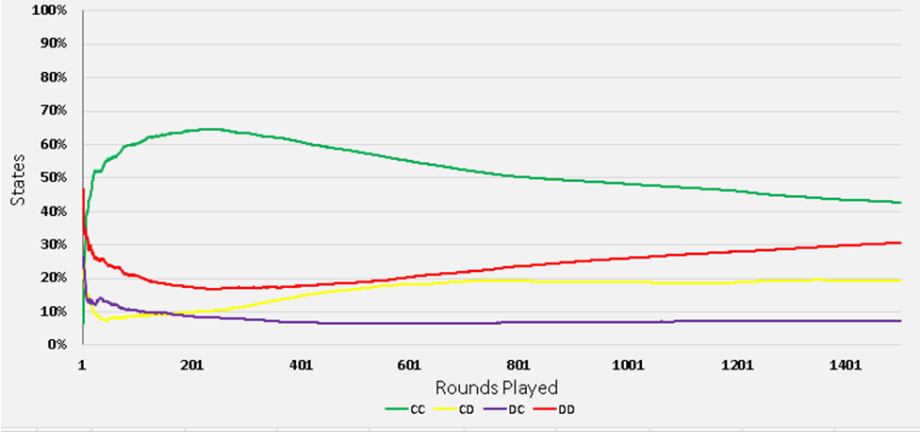

Figure 4.2 b: *Average of the outcomes CC, CD, DC and DD during 1500 rounds of the Q-learning agents playing the IPD game. Constant Payoff Matrix (T=5, R=4 , P=-2 , S=-3) .Self-control states (CC) reach the maximum around 200-300 round then the system starts experiencing a constant decrease on CC states (Ego depletion period)*

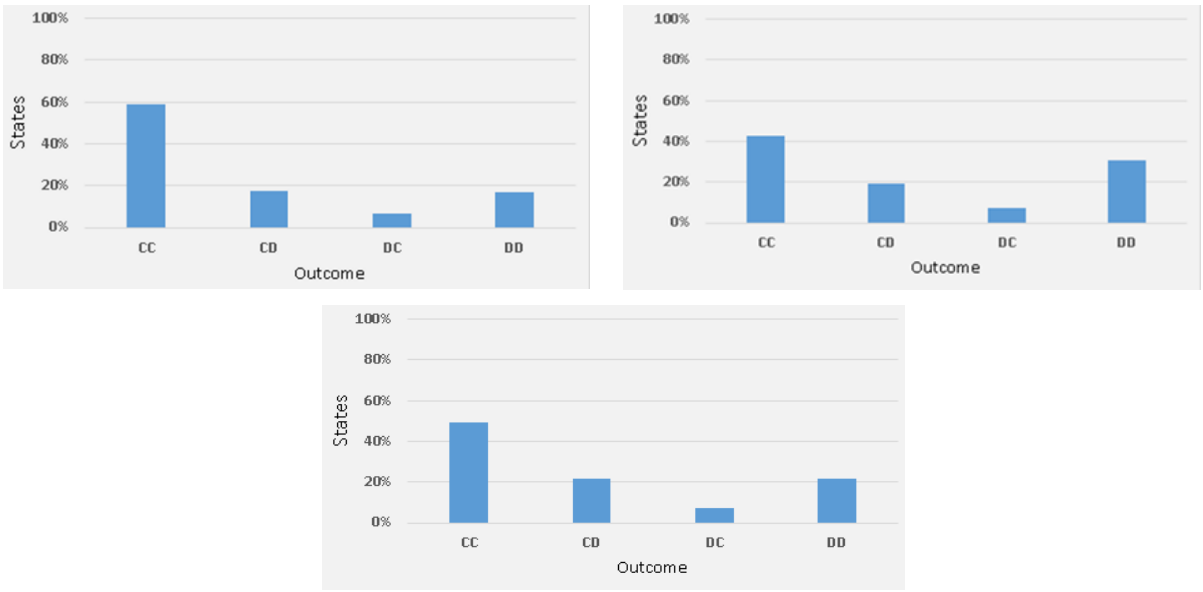

Figure 4.2 c: *Overall average outcomes of the Q-learning agents playing the IPD game. A constant payoff matrix is used: T=5, R=4, P=-2, S=-3. (top-left) Total of rounds = 500 ,(middle) Total of rounds =1000 – baseline results. (top-right) Total of rounds =1500 . Notice how self-control (CC states) decreases as the total of rounds – the duration of the task- increases.*

The phenomenon that we observed could be link to the depletion of self-control resources. Before the system experiences the effects of brain fatigue the organism is restful and thus is more easily driven towards the healthier outcome. However, as the self-control muscle is depleted the organism becomes more vulnerable to temptations and self-control is reduced (Muraven, M. and Baumeister, 2000). We will be referring to the ego depletion period later in our analysis.

4.2.1.2 Increasing the Intensity of Internal conflict

Modelling the intensity of the internal conflict was first suggested by Cleanthous (2010). He proved that by increasing or decreasing the difference between Temptation (T) payoff value and Sucker's (S) payoff value , the internal conflict that the agents experience is more or less intense respectively . In the next experiments we are increasing the intensity of internal conflict using this method in order to explore how long it requires for the system before it reaches the point that we notice a constant decrease in CC states (ego depletion period). We still hold a constant payoff matrix but the starting values for T and S can vary. At the first attempt, we increase the intensity of the conflict by 1 as we set the T to 5.5 and S to -3.5. The other payoff values are set according to the baseline matrix (R = 4, P = -3) and the IPD game last 1000 rounds. We obtained a similar behavior compared to the baseline however, the system reached the maximum in CC states approximately 50 rounds faster than before (Figure 4.2 d). Moving on, in the second experiment we set T to 6.5 and S to -4.5 which increases the intensity of conflict (T-S) by 3. The maximum in CC states has now reached 150 rounds faster compared to the baseline (Figure .4.2 e).

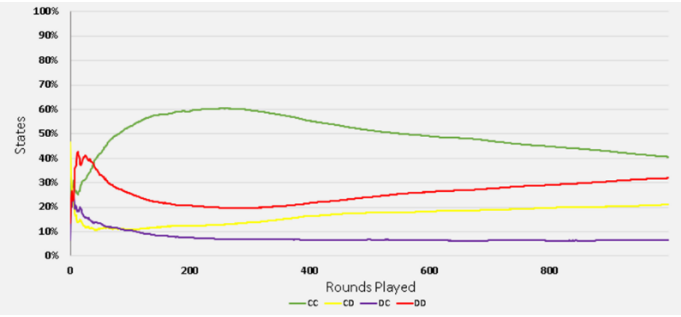

Figure 4.2 d: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Constant Payoff Matrix (T=5.5, R=4 , P=-2 , S=-3.5) ,the intensity of internal conflict (T-S) is increased by 1. Notice that CC states peak at round 250 approximately which is 50 rounds earlier compare to the baseline results (figure 4.2 a). Self-control resources are now depleting faster.*

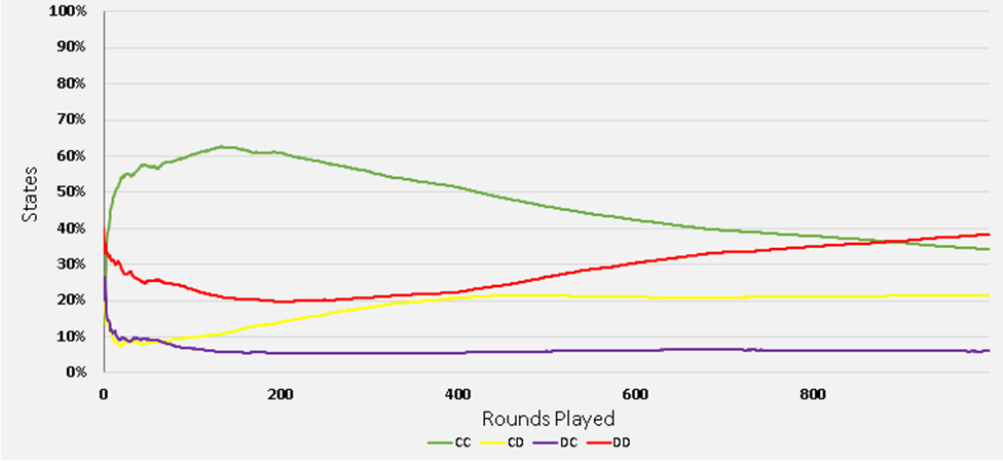

Figure 4.2 e: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Constant Payoff Matrix (T=6.5, R=4 , P=-2 , S=-4.5) ,the intensity of internal conflict (T-S) is increased by 3. Notice that CC states peak at round 150 approximately which is 150 rounds earlier than the baseline results (figure 4.2 a). Self-control resources are now depleted much faster compared to the baseline.*

The previous results indicate that as the intensity of task increases, more self-control resources are required in each round and thus the self-control muscle is depleted faster. This is aligned with Muraven (2000) findings, who showed that participants in tasks that require more self-control are more affected by ego depletion in comparison to when the tasks required less self-control.

4.2.2 Simulating Muscle Recovery

The concept of muscle recovery is implemented in our model with the method *Muscle_Recovery*. The method has two parameters: *recover* variable indicating how many rounds recovery state lasts, *maxDays* variable indicating the interval in rounds before the system enters recovery. The initial payoff matrix is always the payoff matrix of section 4.1.1 and the IPD game lasts 1000 rounds in each trial. At the first attempt, we set the *recover* period to 10 , the *maxDays* interval to 40 and when the agents enter the recovery state T is set to 4.5 and S to -2.5 . The achieved outcome in CC states on the final round was 53% , only 3% higher than the baseline . In the next experiment, we set the *recover* period to 10 and *maxDays* interval to 30 but we keep the same recovery matrix. Again, we have not noticed big difference with the CC states reaching the 54% (figure 4.2 f).

Nevertheless, in both previous experiments the system experiences the effects of self-control fatigue approximately at 300 rounds which is 50 rounds later compared to the baseline. In the third experiment, we tried to increase the recovery period by setting *recover* to 15 and *maxDays* to 30 (figure 4.2 g-h). Self-control was again successfully achieved with the average outcome in CC states reaching the 55%. What was interesting about this experiment, was that not only depletion was delayed for 150 rounds but in addition the maximum CC outcome obtained was 74 % which is 10% more than the baseline maximum.

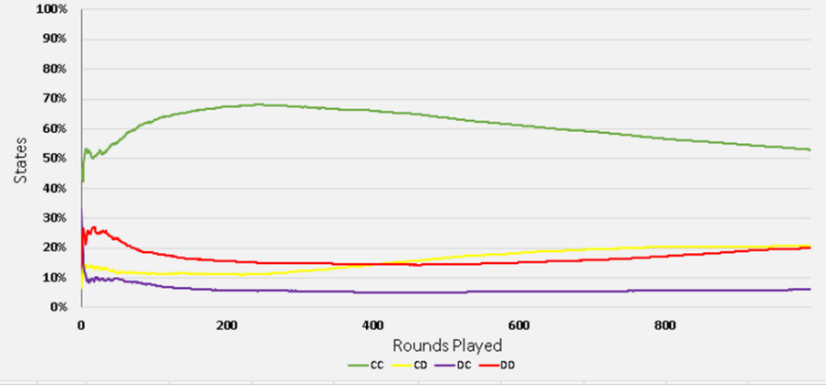

Figure 4.2 f: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Normal Payoff Matrix (T=5.0, R=4 , P=-2 , S=-3) and the Recovery state payoff matrix (T=4.5, R=4 , P=-2 , S=-2.5) . Switching from Normal payoff matrix to Recovery state payoff matrix every 30 rounds and the Recovery state duration is 10 rounds each time. The depletion period is delayed by 50 rounds compared to the baseline and maximum CC states reached are 68%.*

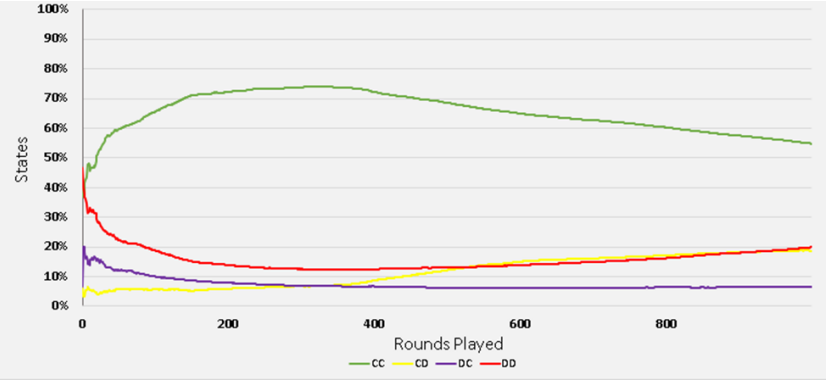

Figure 4.2 g: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Normal Payoff Matrix (T=5.0, R=4 , P=-2 , S=-3) and the Recovery state payoff matrix (T=4.5, R=4 , P=-2 , S=-2.5) . Switching from Normal payoff matrix to Recovery state payoff matrix every 30 rounds and the Recovery state duration is 15 rounds each time. The depletion period is delayed by 150 rounds compared to the baseline and maximum CC states reached are 74%.*

Figure 4.2 h: *Overall average outcomes of the Q-learning agents playing the IPD game. Normal Payoff Matrix (T=5.0, R=4, P=-2, S=-3) and the Recovery state payoff matrix (T=4.5, R=4, P=-2, S=-2.5). (left)Switching from Normal payoff matrix to Recovery state payoff matrix every 30 rounds and the Recovery state duration is 10 rounds each time. (right) Switching from Normal payoff matrix to Recovery state payoff matrix every 30 rounds and the Recovery state duration is 15 rounds each time.*

Moving on, we repeated all 3 previous scenarios but this time the recovery state was set to have T = 4.1 and S=-2.1. In these circumstances, the intensity of the internal conflict of the matrix is the least possible so as to not violate the first rule of the PD game (T > R > P >S). The expected results would be to observe an improvement in self-control compared to the previous experiments due to the decreased conflict of the recovery state. This however was not always correct. When *recover* period was set to 10 and *maxDays* interval to 30 and 40, the achieved self-control was slightly increased than previously for around 1%. Nevertheless, in the last experiment when the *recover* period was set to 15 and *maxDays* interval to 30 the overall average outcome in CC states was 2% less compared to the case that a less intense conflict matrix was used for recovery (figures 4.2i). Furthermore, the maximum outcome in CC states was 67% which is 7% less than before and the overall performance was decreased by 300 (figure 4.2 j).

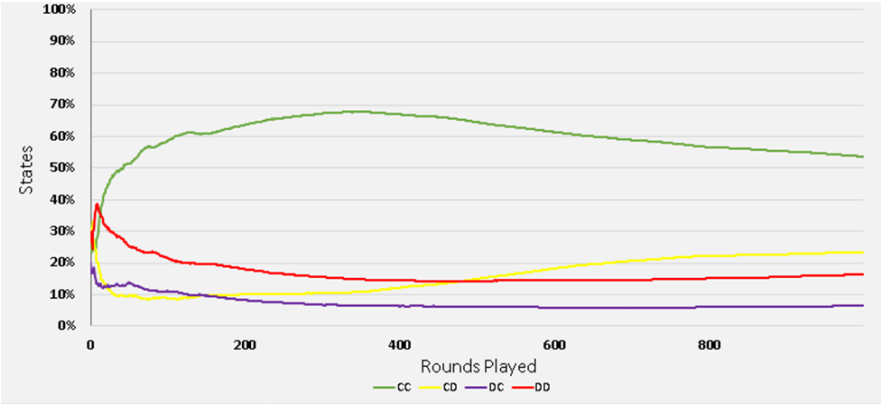

Figure 4.2 i: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Normal Payoff Matrix (T=5.0, R=4 , P=-2 , S=-3) and the Recovery state payoff matrix (T=4.1, R=4 , P=-2 , S=-2.1) . Switching from Normal payoff matrix to Recovery state payoff matrix every 30 rounds and the Recovery state duration is 15 rounds each time. The maximum CC states are 67%.*

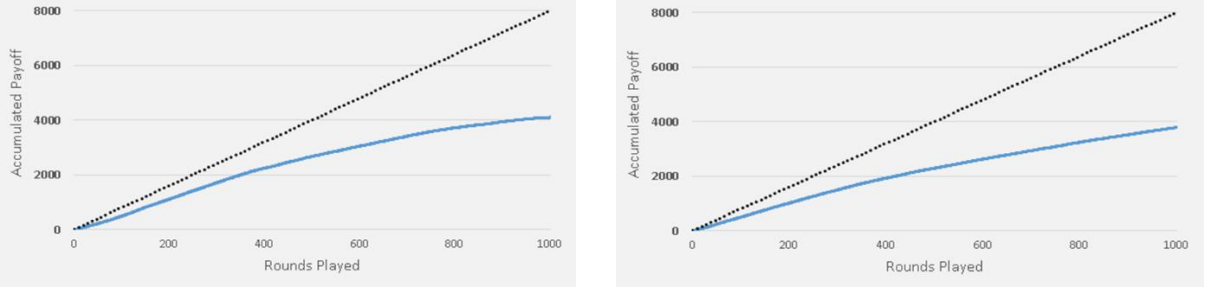

Figure 4.2 j: *Overall performance of the Q-learning agents during the 1000 rounds of the IPD game (blue line). The theoretically best performance is shown for comparison (dot-dashed line). Normal Payoff Matrix values (T=5.0, R=4, P=-2, S=-3). Switching from Normal payoff matrix to Recovery state payoff matrix every 30 rounds and the Recovery state duration is 15 rounds each time. (left) Recovery state payoff matrix values (T=4.5, R=4, P=-2, S=-2.5). (right) Recovery state payoff matrix values (T=4.1, R=4 , P=-2 , S=-2.1) .*

The findings support the muscle recovery theory, and more specifically the de-load resting method[3] since the depleted period was successfully delayed as well as the system managed to achieve a higher maximum performance in self-control during the game. In addition, we detected that too much recovery seems to backfire in some cases.

[3] https://www.muscleandstrength.com/expert-guides/strength

4.2.3 Simulating Task Switching

For the simulations that follow the method *Task_Switching* is being used to discover the behavior in self-control that is accomplished when the system shifts from one task to another (section 3.2). The subsequent tasks could either be of similar or different intensity internal conflict that is defined by the boolean variable *Similar*. The *rounds* parameter determines the interval (number of rounds) before shifting to a new task. In all cases, the initial payoff matrix that is used is the same as the baseline matrix (section 4.1.1).

4.2.3.1 Switching to Similar Intensity conflict tasks

At the first attempt, we set the *rounds* variable to 500 which means we will be shifting only to 1 subsequent task of similar internal conflict at the round 500 of IPD game during each trial. Although, self-control was achieved the overall average CC states that we obtain was 53% that is only 3% higher compared to the baseline results (figure 4.2k). The influence of one subsequent task later in the game did not show significant impact on the desired outcome. A similar outcome was accomplished when we increased the subsequent tasks to 3 by setting the *rounds* variable interval to 250. It appears that when we use large intervals for task switching self-control behavior was not essentially reinforced (figure 4.2.L).

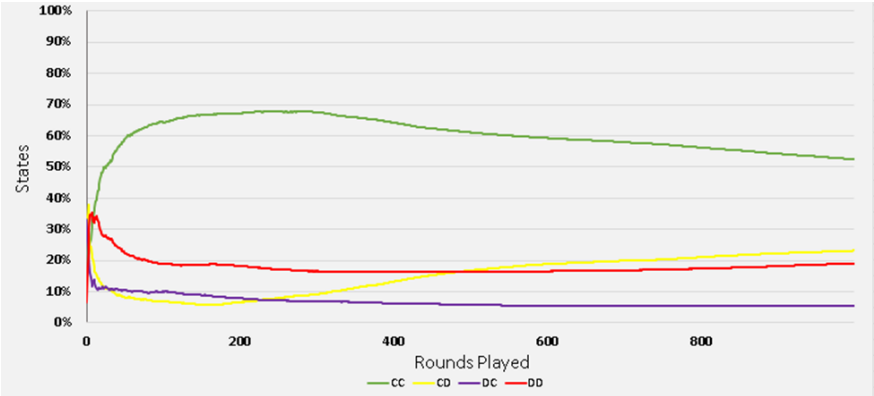

Figure 4.2 k: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Initial Payoff Matrix (T=5, R=4, P=-2, S=-3). Switching to a random Similar Payoff Matrix every 500 rounds.*

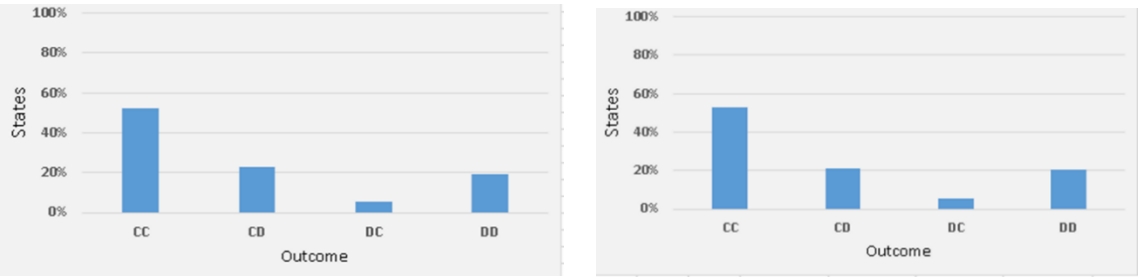

Figure 4.2 L: *Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game. Initial Payoff Matrix (T=5, R=4 , P=-2 , S=-3) . (left) Switching to a random Similar payoff Matrix every 500 rounds. (right) Switching to a random Similar payoff Matrix every 250 rounds. Both simulations have a similar outcome.*

We then tried smaller intervals for task switching with the aim of achieving a higher impact on self-control states. In the next experiment we set the interval to 125, which equals to 7 subsequent tasks during the IPD game. The outcome in CC states was 58%, around 8% more compared to the baseline results (figure 4.2 m). The best performance was achieved when the interval was set to 100, with CC states reaching 62% (figure 4.2n). In all the previous scenarios, self-control was indeed improved when the subsequent task involved a similar internal conflict. The organism seems to perform better when the game involves many continuous tasks with smaller duration.

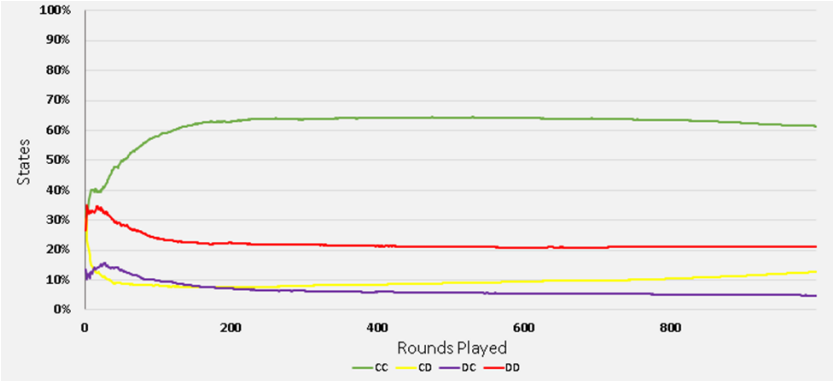

Figure 4.2 m: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Initial Payoff Matrix (T=5, R=4 , P=-2 , S=-3) . Switching to a random Similar Payoff Matrix every 100 rounds.*

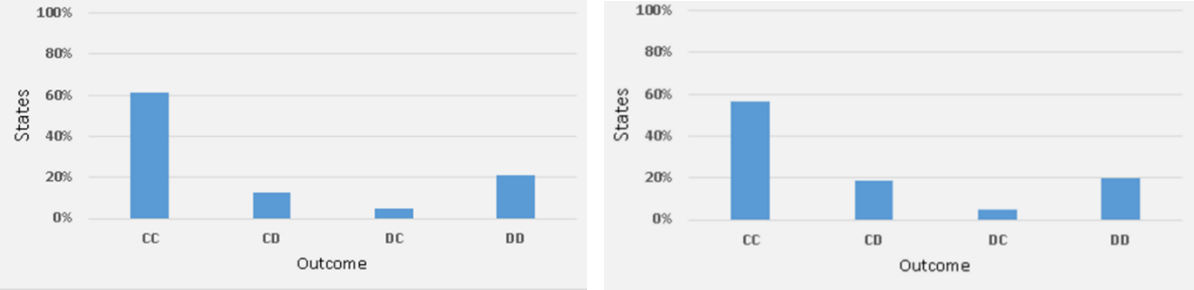

Figure 4.2 n: *Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game. Initial Payoff Matrix (T=5, R=4 , P=-2 , S=-3) . (left) Switching to a random Similar payoff Matrix every 100 rounds. (right) Switching to a random Similar payoff Matrix every 125 rounds. Self-control was crucially increased in both cases.*

4.2.3.2 Switching to Different Intensity conflict tasks

A series of similar experiments was performed to discover the influence in self-control when the subsequent task involved a different internal conflict. In the first experiment, we set the *rounds* variable to 500 to shift only to 1 subsequent task during each trial. We would expect to view a small negative impact on CC states, since no significant change was observed in the previous scenario with a similar intensity subsequent task. Surprisingly enough, not only self-control was not achieved but the overall average CC states we obtained was 42%, which is 8% lower than the baseline results (figure 4.2o). The CC states were reduced by 32% when *rounds* variable interval was set to 250 and DD states reached the 40% (figure 4.2p). In contrast with the scenario that a similar conflict subsequent task was used, we now see a huge impact on self-control even with large interval values (figures 4.2q).

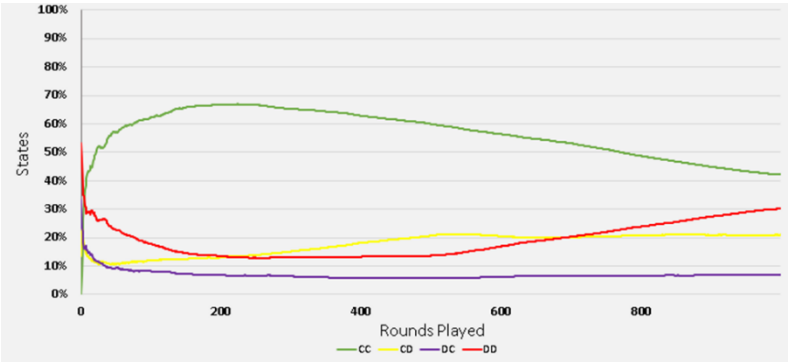

Figure 4.2 o: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Initial Payoff Matrix (T=5, R=4 , P=-2 , S=-3) . Switching to a random Different intensity payoff matrix every 500 rounds.*

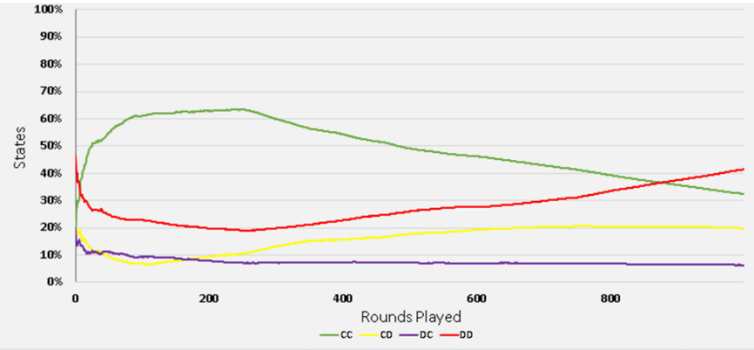

Figure 4.2 p: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Initial Payoff Matrix (T=5, R=4 , P=-2 , S=-3) . Switching to a random Different intensity payoff matrix every 250 rounds.*

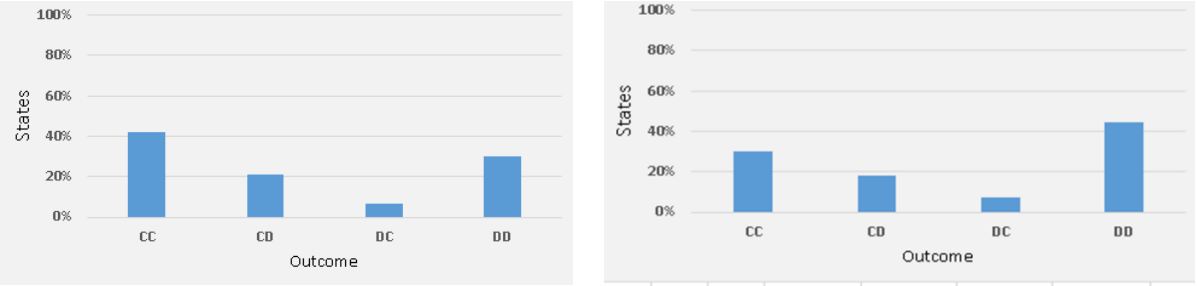

Figure 4.2 q: *Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game. Initial Payoff Matrix (T=5, R=4 , P=-2 , S=-3) . (left) Switching to a random Different intensity payoff matrix every 500 rounds. (right) Switching to a random Similar payoff Matrix every 250 rounds. Notice the impact on CC and DD states.*

Finally, self-control was further decreased when smaller interval values were used. We set the *rounds* interval to 100 and 125, and the achieved outcome in CC states was only 29% and 31%, respectively (figure 4.2 r). The difference in the overall performance when large or small *rounds* interval is used is presented in figure 4.2 s. The earlier results support the findings of the studies done by Dewitte, Bruyneel and Geyskens (2009) related to the consequence in self-control when an initial self-control task is followed by subsequent tasks of similar or different response conflict.

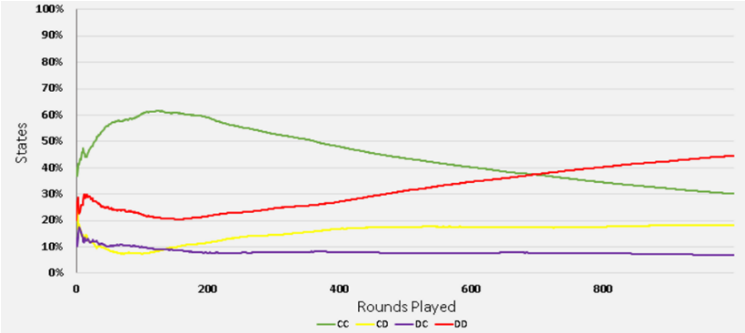

Figure 4.2 r: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Initial Payoff Matrix (T=5, R=4 , P=-2 , S=-3) . Switching to a random Different intensity payoff matrix every 100 rounds.*

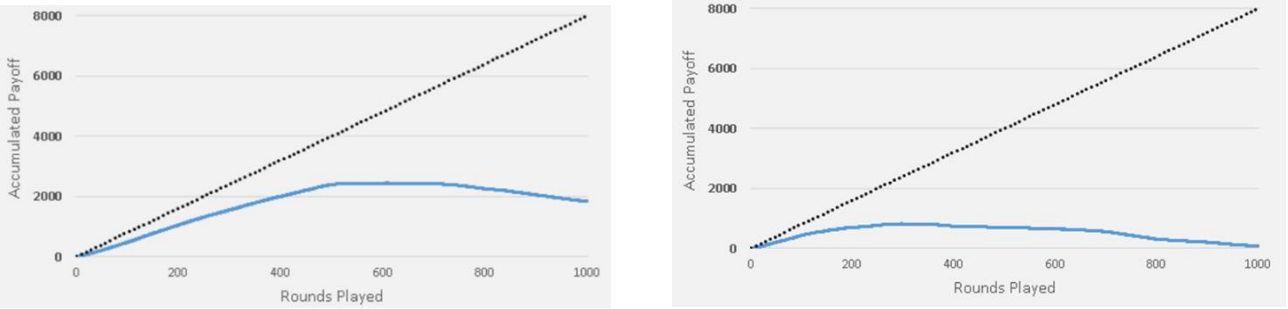

Figure 4.2 s: *Overall performance of the Q-learning agents during the 1000 rounds of the IPD game (blue line). The theoretically best performance is shown for comparison (dot-dashed line). Initial Payoff Matrix (T=5, R=4 , P=-2 , S=3). (left) Switching to a random Different intensity payoff matrix every 500 rounds. (right) Switching to a random Different intensity payoff matrix every 100 rounds. Self-control was not achieved in both scenarios.*

4.2.4 Summary and discussion on Self-control Strength model

The self-control strength model theory entails that self-control operates like a muscle (Muraven and Baumeister,2000). The first concept we manage to observe from our model was the depletion of the self-control muscle that appears during a self-control task. From our simulations, we observed that during the IPD game the system had an upward trend towards the behavior of self-control, then CC states reached a maximum point. Right after that there was downward trend with the self-control constantly decreasing. Despite the system's ability to converge towards the most beneficial outcome at the beginning, when self-control resources were crucially depleted the system became more vulnerable to temptations and thus self-control was decreasing.

The muscle or ego depletion period appeared earlier during the game when higher conflict matrix was used. Moving on, by incorporating the concept of muscle recovery in our model we observed some interesting facts. Not only the ego depletion period was delayed during the IPD game but in addition the organism was able to perform a higher overall performance. These findings align with the overloading resting method, and higher performance that could be achieved when muscles enter supercompensation phase (see section 2.3.1).

Finally, we deployed the concept of self-control task switching in our model. Dewitte, Bruyneel and Geyskens (2009) showed through their studies that self-control performance increases and decreases when an initial self-control task is followed by a subsequent task of similar or different response conflict, respectively. The impact of a single subsequent tasks of similar conflict was not crucial on the increased self-control. However, when multiple subsequent tasks of similar conflict were presented in the system, we attained a huge improvement on self-control. When the system was switching to different response conflict tasks there was a significant decrease in performance despite the task shifting interval that was used.

4.3 Extrinsic Motivation

Extrinsic motivation refers to behavior that is driven by external factors such as a reward or a negative outcome (Berlyne, 1966). The concept is implemented in our model with the methods *External_Monetary_Reward* and *External_Motivator*. The first method is related to an external reward that arrives during the IPD game in order to influence positively the system when depletion is experienced. The second method is associated to an external condition or authority that changes the rules of the game, by forcing the two agents to periodically choose to cooperate. For all the following experiments we will be using the same constant payoff matrix as our baseline model (section 4.1.1) and the IPD game lasts 1000 rounds in each trial.

4.3.1 Simulating External monetary reward

The method has one parameter as input, the variable *discount_interval.* This value defines the exact round at which the external reward is presented in the system, which causes the lower brain agent discount rate to decrease by 0.3. At the first attempt, we set the *discount_interval* value to 750. The overall average in CC states we attained was 53%, which is only 3% higher than the baseline (figure 4.3 a). As we previously discuss in section 4.2.1.1, the depletion period is experienced approximately after 300 rounds during the game. The external reward was presented too late compared to the depletion period and due to the lack of self-control resources the system could not change behavior at 750 rounds. In the next experiment we set the *discount_interval* to 500, that is much closer to the depletion starting point (figure 4.3 b-c). The obtained CC states outcome was around 57%.

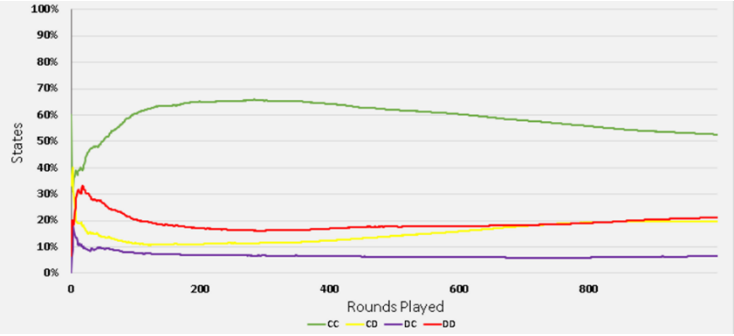

Figure 4.3 a: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Constant Payoff Matrix (T=5, R=4, P=-2, S=-3). At round 750 the discount rate of the lower brain agent decreases by 0.3.*

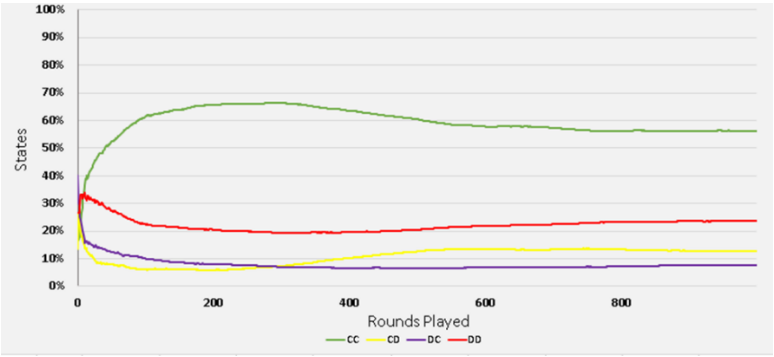

Figure 4.3 b: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Constant Payoff Matrix (T=5, R=4 , P=-2 , S=-3) . At round 500 the discount rate of the lower brain agent decreases by 0.3.*

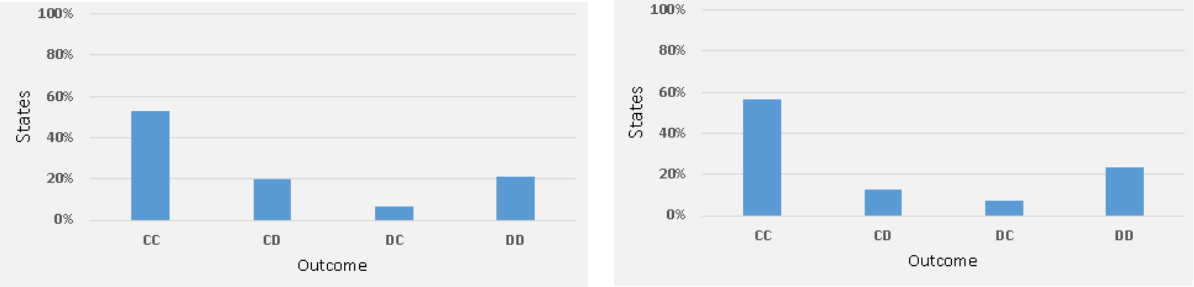

Figure 4.3 c: *Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game. Constant payoff Matrix (T=5, R=4 , P=-2 , S=-3). (left) Discount interval round set to 750. (right) Discount interval round set to 500.*

In the final experiment we set the *discount_interval* to 300 rounds, almost at the depletion starting point. With the final attempt we achieved the best performance with the CC states reaching 61% (figures 4.3 d-e). The system did not experience fatigue during the game and also managed to perfectly sustain the desired outcome in a very high percentage. Additionally, the accumulated payoff was significantly improved and approached the theoretical best payoff.

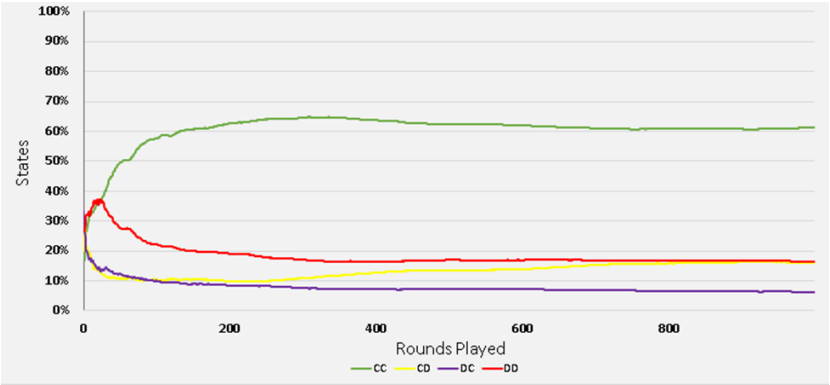

Figure 4.3 d: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Constant Payoff Matrix (T=5, R=4 , P=-2 , S=-3) . At round 300 the discount rate of the lower brain agent decreases by 0.3 . Notice how the CC states are sustained on high percentages.*

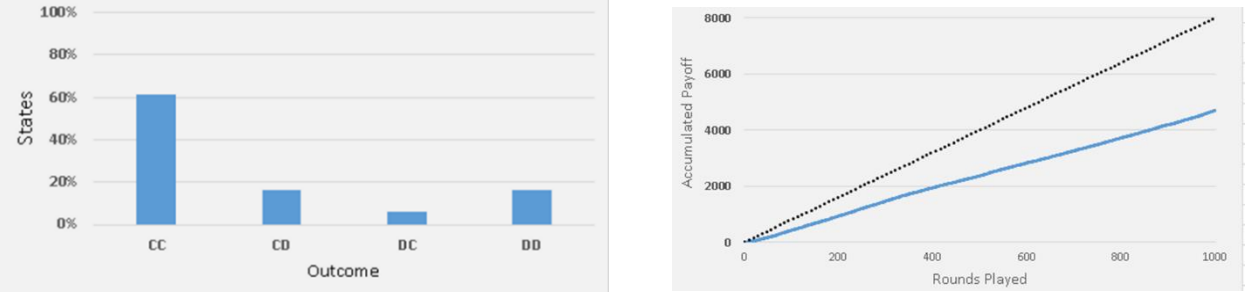

Figure 4.3 e: *Constant payoff matrix (T=5, R=4 , P=-2 , S=-3) .Discount interval round set to 300. (left) Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game. (right) Overall performance of the Q-learning agents during the 1000 rounds of the IPD game (blue line). The theoretically best performance is shown for comparison (dot-dashed line). The achieved performance is 1000 points greater than the baseline results.*

4.3.2 Simulating External motivator

The next method has one parameter that is the variable *forced_rounds_interval* which defines how often the agents are forced in the game to cooperate. In the first experiment, we set that interval to 50. The attained results indicate that there was no difference in the CC states compared to the baseline results (figure 4.3 f). Obviously, the interval that the agents are forced to cooperate is not enough so as to change their overall strategy for the rest of the game.

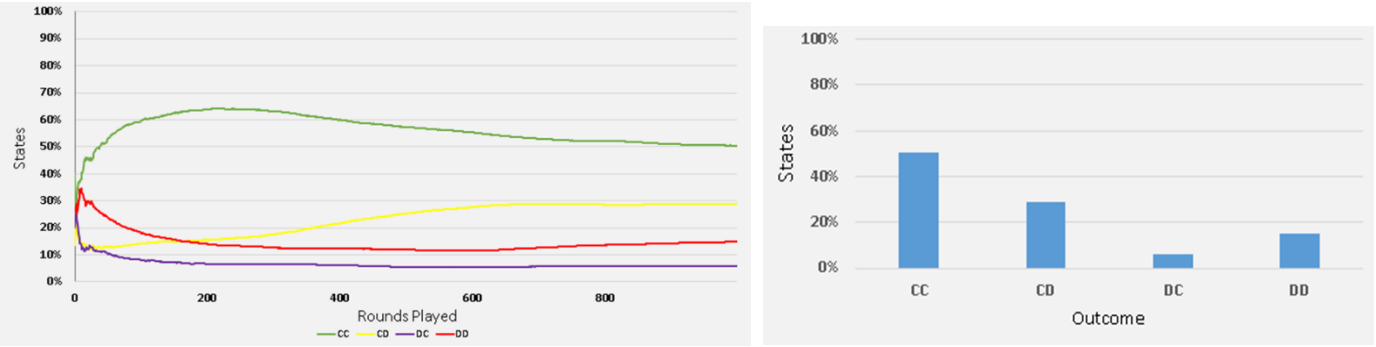

Figure 4.3 f: *Constant Payoff Matrix (T=5, R=4 , P=-2 , S=-3) and forced rounds interval is set to 50.(left) Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game.(right)Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game . The achieved self-control (CC) states did not change compared to the baseline.*

In the second experiment, we decrease the *forced_rounds_interval* to 25. With a much smaller interval we expect to see improved performance. That was indeed accurate since the overall average outcome of CC states was 58%, that is 8% higher compared to the baseline (figure 4.3 g-h). In addition, the attained accumulated payoff increased by 1000 compared to the baseline (figure 4.3h).

.

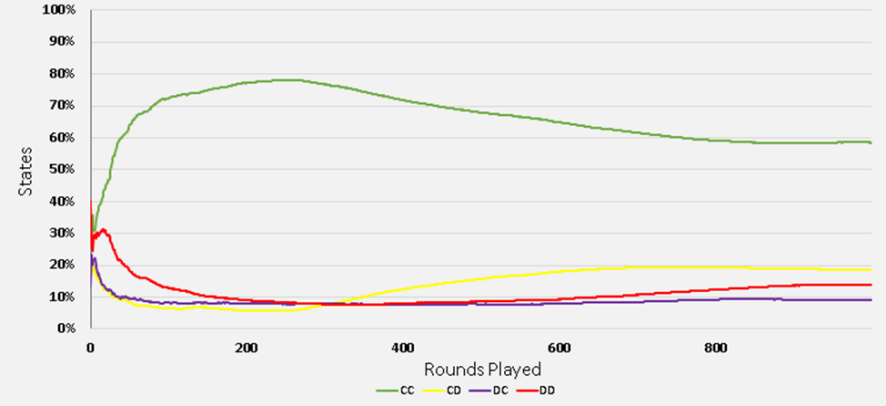

Figure 4.3 g: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Constant Payoff Matrix (T=5, R=4 , P=-2 , S=-3) .Forced rounds interval is set to 25. The maximum CC states reached was 78%.*

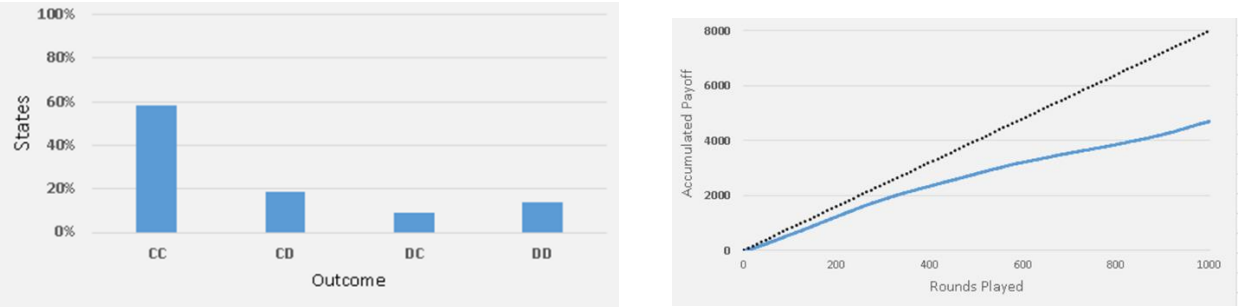

Figure 4.3 h: *Constant payoff matrix (T=5, R=4 , P=-2 , S=-3) .Forced rounds interval round set to 25. (left) Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game. (right) Overall performance of the Q-learning agents during the 1000 rounds of the IPD game (blue line). The theoretically best performance is shown for comparison (dot-dashed line). The achieved performance is 1000 points higher than the baseline results and self-control outcome states was increased.*

The best outcome was obtained by setting the interval to 10, where CC states reached 63% (figures 4.3.i-j) . It is worth mentioning that by using smaller intervals we managed to achieve at a certain round CC states around 80%. Furthermore, the the overall accumulated payoff improved and undoubtedly approaches the theoretical best performance. The previous findings reflect how crucial is the influence of an external player when it comes to general sum games. Although it is still tempting in any round for the agents to deflect, the system is driven by the external motivator towards the best strategy which is the mutual cooperation.

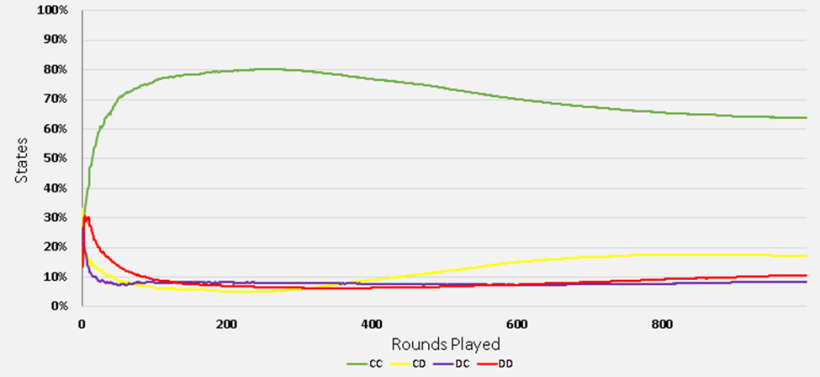

Figure 4.3 i: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Constant Payoff Matrix (T=5, R=4 , P=-2 , S=-3) .Forced rounds interval is set to 10. The maximum outcome in CC states was 80%.*

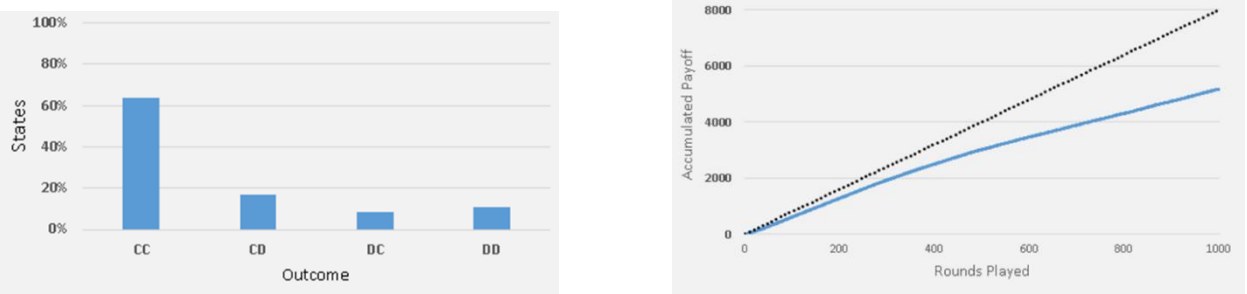

Figure 4.3 j: *Constant payoff matrix (T=5, R=4 , P=-2 , S=-3) .Forced rounds interval round set to 10. (left) Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game. (right) Overall performance of the Q-learning agents during the 1000 rounds of the IPD game (blue line). The theoretically best performance is shown for comparison (dot-dashed line). The achieved performance is 1500 points higher than the baseline results and self-control states outcome increased by 12%.*

4.3.3 Summary and discussion on Extrinsic motivation

Extrinsic motivation is when someone or something is conditioned to behave a certain way due to a reward or consequence (Berlyne, 1966). In our simulations, we deployed the concept of the appearance of an external reward during the IPD game. The round in which the external reward was presented in the system defined how much self-control was improved. When the external reward was presented later in the game, there was not major impact on self-control. However, when it showed up at a round near the peak in CC states, we documented a major improvement on self-control with the system managing to sustain the desired outcome to high percentages for the rest of the game. The elimination of the depletion effect was indeed accomplished by the offered the monetary incentive (Baumeister, 2006).

Moving on, we deployed an external motivator which forced the two agents to mutual cooperation periodically. When the interval for the forced rounds was large, we did not notice a change in the overall performance. Despite of that, with smaller intervals the system was driven towards cooperation more often and that forced the two agents to learn to follow the most beneficial strategy for system. The maximum CC states we attained with this method was around 80%.

4.4 Simulating Self-control as a virtue

For the last simulations we are using the method *Clear_Standards* that is associated to the self-control virtue theory. The presence of negative emotions is experience in our model by periodically decreasing or increasing the R and T payoff values, respectively. We will discover in the following experiments how self-control is influenced when we periodically reset the payoff matrix to the initial " standard " values. The concept of self-control virtue is analysed in more detail in section 3.4. The initial payoff matrix values are T=5 , R=4 , P=-2 , S=-3 and the IPD game last 1000 rounds .

4.4.1 Presence of Negative emotions

We first need to define the baseline when only the presence of negative emotions is experienced in the model. The presence of negative emotions was first implemented by Nikodemou(2020) and is explained in more detail along with the virtue theory in section 2.4.We start by setting the payoff interval to 25 and the ratio that R change to -0.1 . That means in every 25 rounds during the IPD game the R payoff value will decrease by 0.1. The results we obtain are presented in figures 4.4a-b. Self-control was not achieved and the overall average outcome in CC state was 32%. We will later use these results as the baseline for the experiments where only R changes . The second baseline results we will be using are attained when the payoff interval is set to 25 and the ratio that T changes is set to 0.1 (figures 4.4c-d). The overall average outcome in CC states was 38 %. The third baseline results are obtained when both T and R are increasing or decreasing respectively, with payoff interval 25 and change ratio 0.1 (figures 4.4e-f). The overall average obtain outcome in CC states for the last scenarios was 26% and DD states reached 41%. In all three cases the two agents did not learn to cooperate.

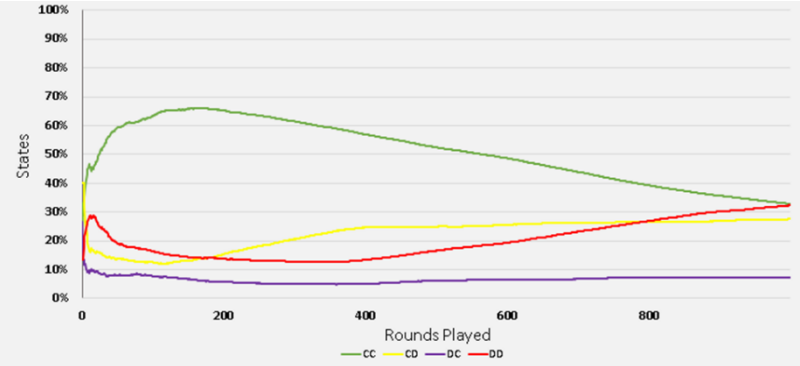

Figure 4.4 a: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Decreasing the R payoff. Ratio R=0.1 and payoff interval=25. This is the baseline when only R changes.*

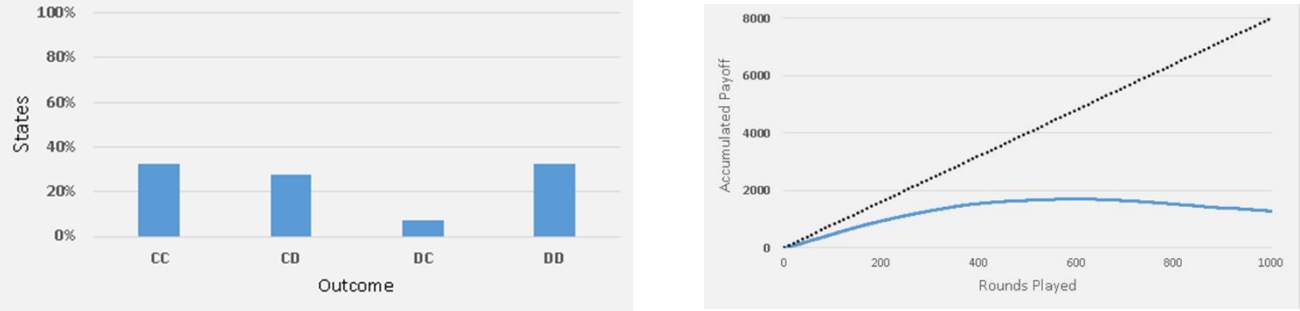

Figure 4.4 b: *Initial payoff matrix (T=5, R=4 , P=-2 , S=-3) . Decreasing the R payoff. Ratio R=0.1 and payoff interval=25. (left) Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game. (right) Overall performance of the Q-learning agents during the 1000 rounds of the IPD game (blue line). The theoretically best performance is shown for comparison (dot-dashed line)*

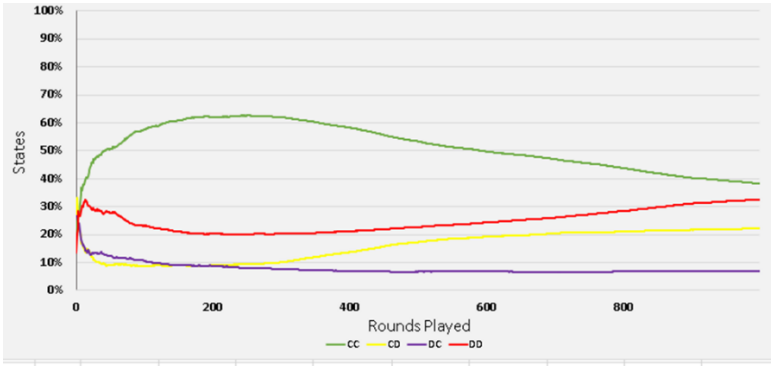

Figure 4.4 c: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Increasing the T payoff. Ratio T=0.1 and payoff interval=25. This is the baseline when only T changes.*

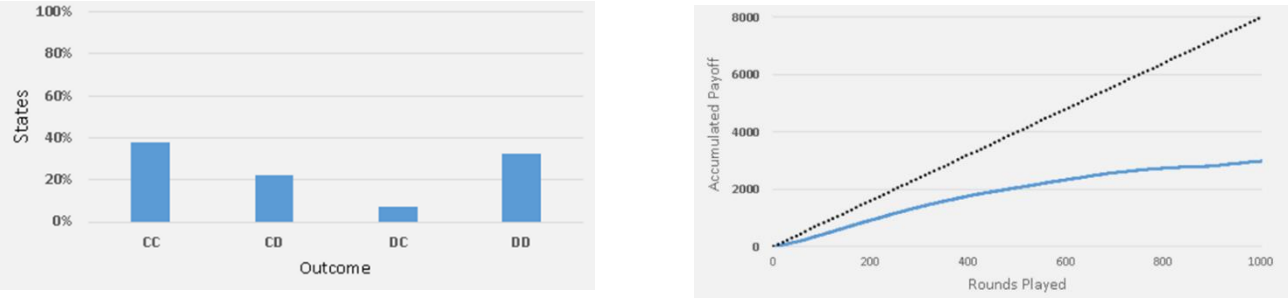

Figure 4.4 d: *Initial payoff matrix (T=5, R=4 , P=-2 , S=-3) . Increasing the T payoff. Ratio T=0.1 and payoff interval=25. (left) Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game. (right) Overall performance of the Q-learning agents during the 1000 rounds of the IPD game (blue line). The theoretically best performance is shown for comparison (dot-dashed line)*

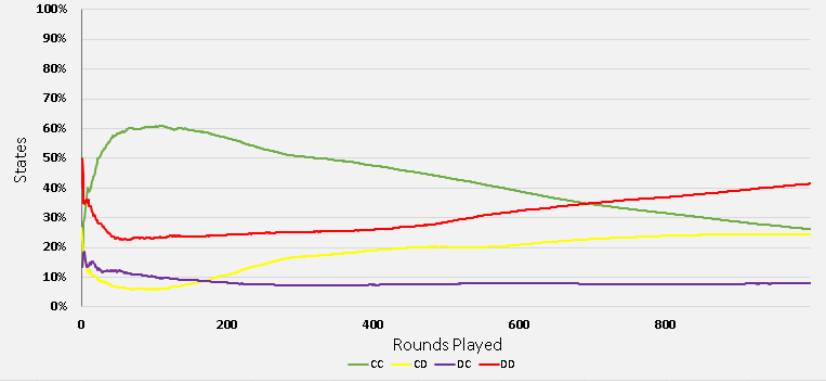

Figure 4.4 e: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Increasing the T payoff and decreasing R payoff. Ratio T=0.1,R=0.1 and payoff interval=25.This is the baseline when both T and R changes.*

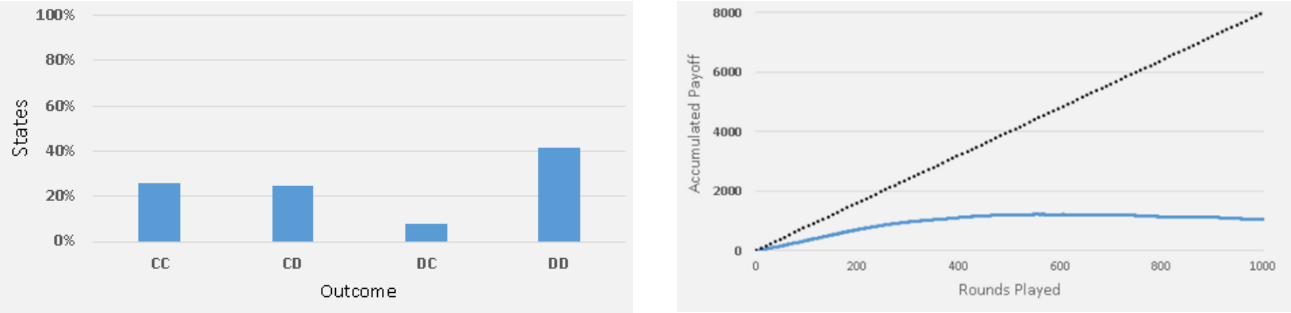

Figure 4.4 f: *Initial payoff matrix (T=5, R=4 , P=-2 , S=-3) . Increasing the T and decreasing R payoff . Ratio T=0.1,R=0.1 and payoff interval=25.(left) Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game. (right) Overall performance of the Q-learning agents during the 1000 rounds of the IPD game (blue line). The theoretically best performance is shown for comparison (dot-dashed line)*

4.4.2 Simulating Clear standards

Having used the presence of negative emotions in 4.4.1 that produced the baseline results, we are now ready to experiment using the *Clear_Standards* method. We will examine how self-control is influenced by incorporating this method when R and T payoff values change separately or together when we keep the baseline parameters for payoff ratio and interval. In all scenarios the IPD game lasts 1000 rounds.

4.4.2.1 Decreasing the Reward payoff

In the following experiments, we always keep the R ratio to -0.1, the payoff interval to 25 but we are changing the standards interval. At the first experiment, the standards interval was set to 250 rounds. This means in every 25 rounds the R payoff value will decrease by 0.1 and in every 250 rounds all the payoff values along with R will reset to the initial values. The overall average outcome in CC states was 45% (figure 4.4 g). Although self-control was not successfully achieved, the obtain outcome was 13% higher compared to the baseline (figures 4.4 a-b). The overall performance was also increased with the achieved accumulated payoff value being 1400 greater than the baseline result.

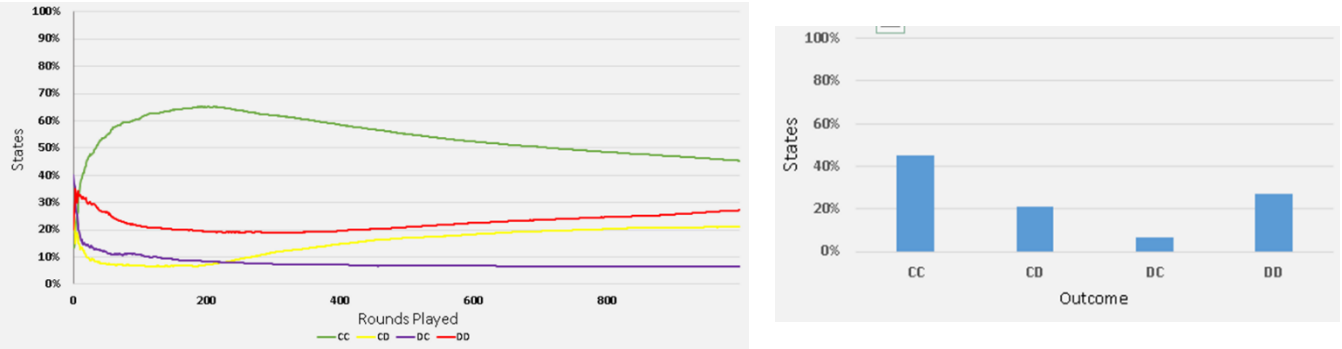

Figure 4.4 g: *Initial Payoff Matrix (T=5, R=4 , P=-2 , S=-3) . Decreasing R payoff. Ratio R=0.1 and payoff interval=25 . The system reset to the initial payoff matrix (T=5 ,R=4 ,P= -2. S= -3) every 250 rounds (standards interval = 250). (left) Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. (right)Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game. The achieved self-control (CC) states were 13% compared to the baseline.*

Moving on, in the next experiment we used a smaller interval for the standards reset. When standards interval was set to 125 not only the desired behaviour was successfully achieved with CC states reaching the 50% but also the overall accumulated performance of the system increased by 2200 (figures 4.4 h-j). It is noteworthy that even when we set the standards interval to 500, a much greater value, the produced CC states was again increased by around 9%. Despite the constant presence of negative with the R (Reward) payoff value periodically decreasing, the capability of the system to often change behaviour according to the relevant standards had a significant impact on the overall outcome.

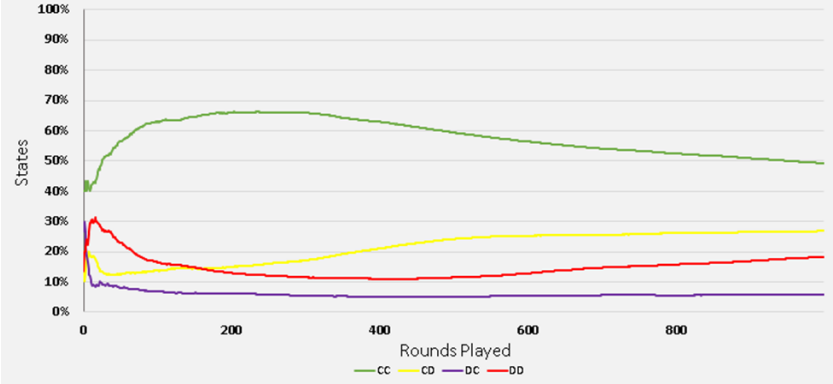

Figure 4.4 h: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Decreasing R payoff. Ratio R=0.1 and payoff interval=25. The system reset to the initial payoff matrix (T=5 ,R=4 ,P= -2 . S= -3) every 125 rounds (standards interval = 125)*

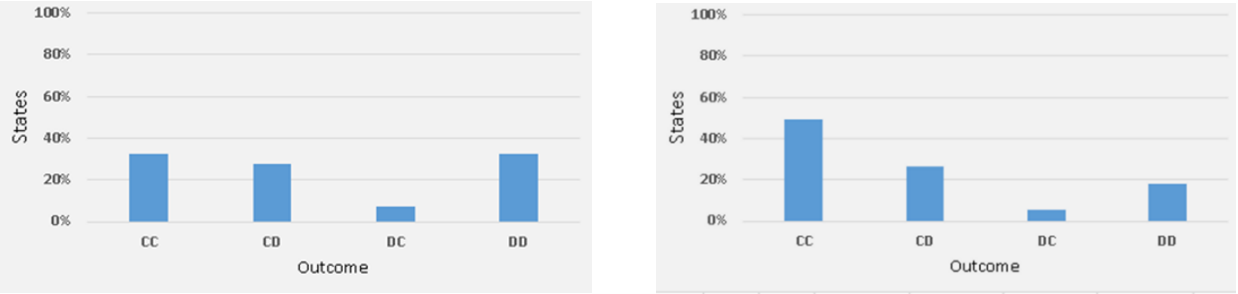

Figure 4.4 i: *Initial payoff matrix (T=5, R=4 , P=-2 , S=-3) . Decreasing the R payoff. Ratio R=0.1 and payoff interval=25. (left) Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game (baseline). (right) Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game with standards interval = 125.*

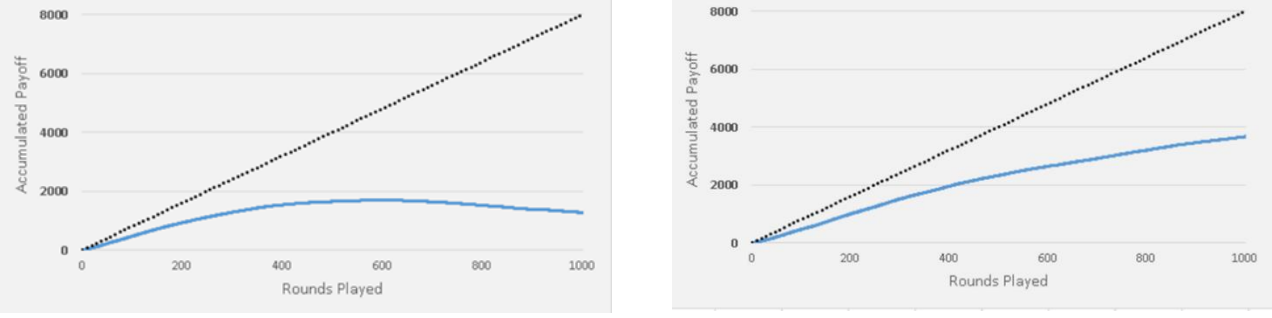

Figure 4.4 j: *Overall performance of the Q-learning agents during the 1000 rounds of the IPD game (blue line). The theoretically best performance is shown for comparison (dot-dashed line). Initial Payoff Matrix (T=5, R=4 , P=-2 , S=3). Decreasing the R payoff. Ratio R=0.1 and payoff interval=25. (left) Baseline results. (right) Standards interval set to 125. Strong impact on the performance indicating that the system sustains the desired behaviour.*

4.4.2.2 Increasing the Temptation payoff

A series of similar experiments was performed with only the T(Temptation) payoff value increasing periodically every 25 rounds. We would expect to encounter an analogous improvement on self-control by incorporating the same concept. When the standards interval was set to 500, we did not observe a huge impact on self-control since the CC states appear only 42% of the time. Moving on, we tested the method with smaller standards interval. We set the standards interval to 250 and 125, and the overall average outcome in CC states was 45% and 47% respectively (figure 4.4 k).

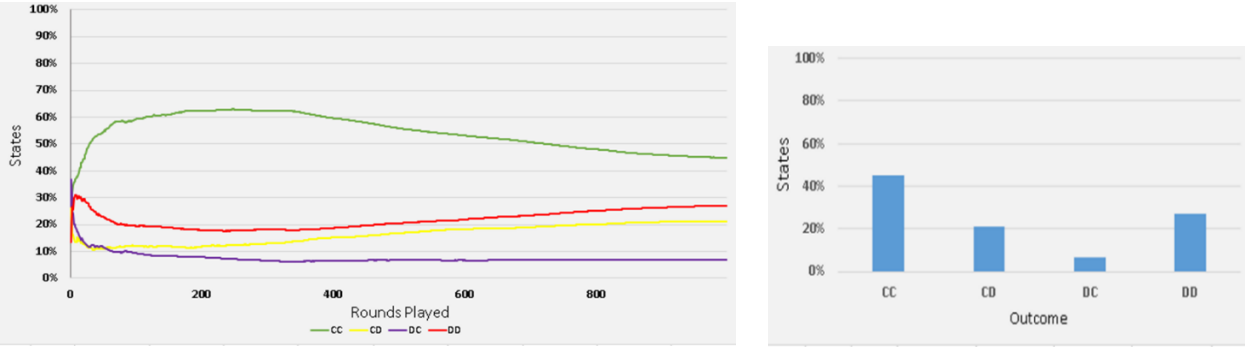

Figure 4.4 k: *Initial Payoff Matrix (T=5, R=4 , P=-2 , S=-3) . Increasing T payoff. Ratio T=0.1 and payoff interval=25 . The system reset to the initial payoff matrix (T=5 ,R=4 ,P= -2 . S= -3) every 250 rounds (standards interval = 250).(left) Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game.(right)Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game . The achieved self-control (CC) states were 7% compared to the baseline.*

The best outcome was obtained with standards interval set to 100. The overall attained CC states was 50% which is 12% higher compared to the baseline (figures 4.4 l-m). Furthermore, the accumulated performance in the last scenario was also increased by 1000 (figure 4.4 n). The *clear_standards* method did indeed influence positively self-control, however smaller intervals were required in order to view a meaningful impact.

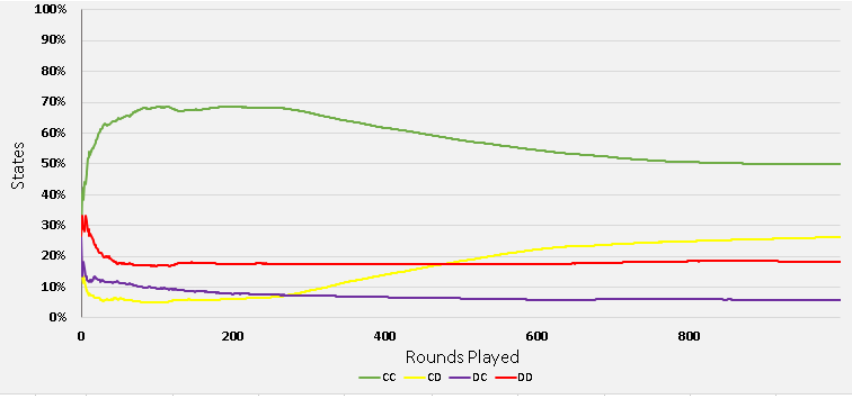

Figure 4.4 l: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Increasing T payoff. Ratio T=0.1 and payoff interval=25 . The system reset to the initial payoff matrix (T=5 ,R=4 ,P= -2 . S= -3) every 100 rounds (standards interval = 100)*

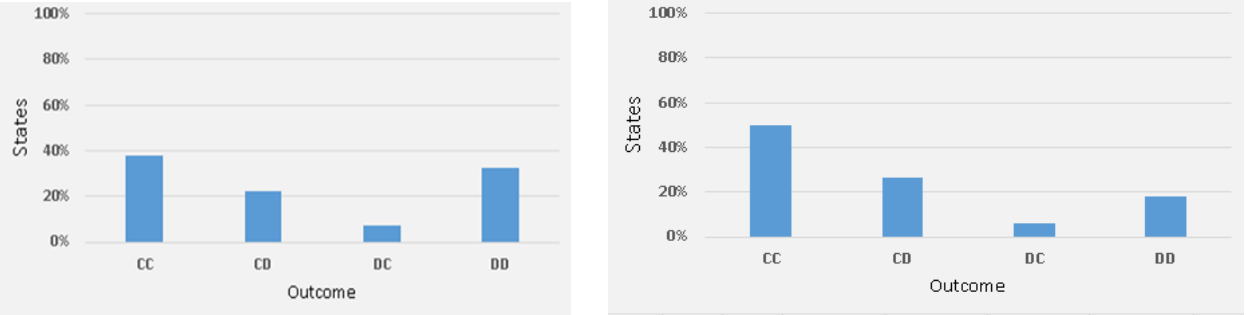

Figure 4.4 m: *Initial payoff matrix (T=5, R=4 , P=-2 , S=-3) . Increasing the T payoff. Ratio R=0.1 and payoff interval=25. (left) Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game (baseline) .(right) Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game with standards interval = 100.*

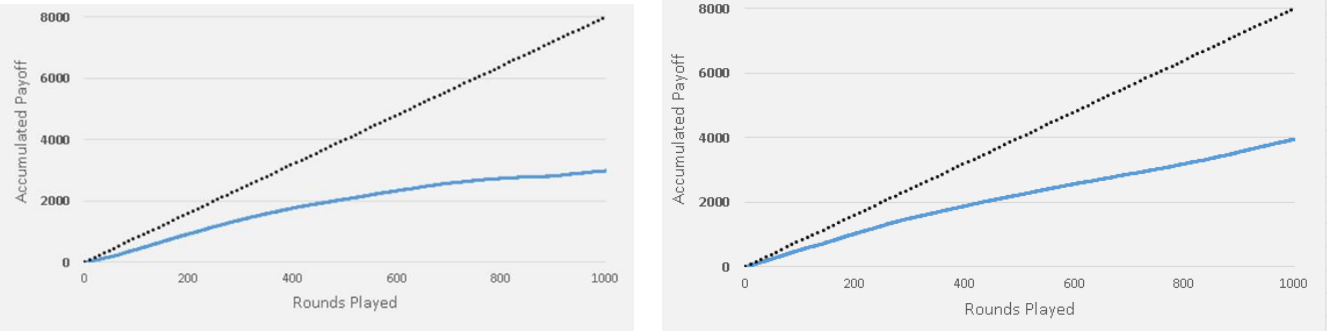

Figure 4.4 n: *Overall performance of the Q-learning agents during the 1000 rounds of the IPD game (blue line). The theoretically best performance is shown for comparison (dot-dashed line). Initial Payoff Matrix (T=5, R=4 , P=-2 , S=3). Increasing the T payoff. Ratio T=0.1 and payoff interval=25 . (left) Baseline results . (right) Standards interval set to 100.*

4.4.2.3 Increasing Temptation and Decreasing Reward payoff

In the final experiments, we discover how well our method performs when there is the presence of an intense negative emotion. More specifically, we will examine the influence of the method when both R(Reward) and T(Temptation) payoff values are decreasing and increasing respectively with a payoff interval 25 and a payoff ratio 0.1. At first, we set the standards interval to 500, which means in every 25 rounds of the IPD game the R and T payoff values are decreased and increased by 0.1 respectively, and in every 500 rounds all the payoff values are reset to the initial payoff values (T=5, R=4, P=-2, S=-3). The produced CC states were 32% which is only 6% greater than our baseline results (figures 4.4 e-f and figure 4.4 o). There was neither a huge improvement on self-control when the standards interval was set to 250.

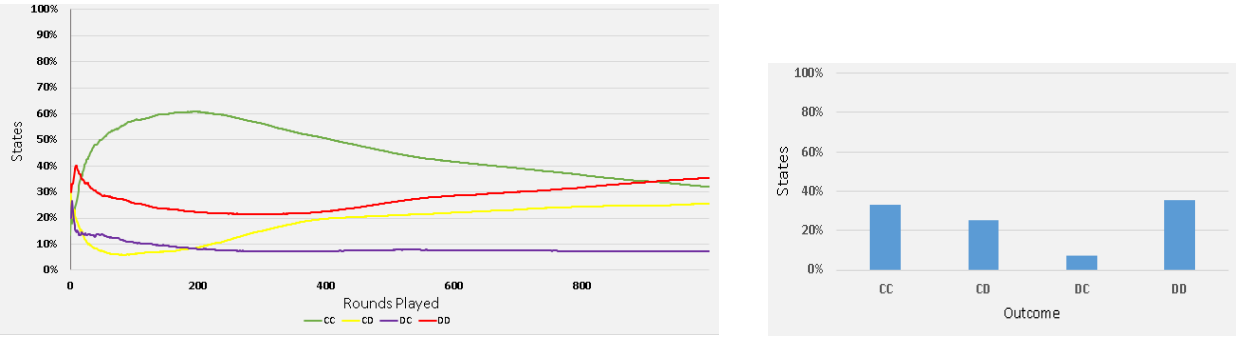

Figure 4.4 o: *Initial Payoff Matrix (T=5, R=4 , P=-2 , S=-3) . Increasing T and decreasing R payoff. Ratio T=0.1,R=0.1 and payoff interval=25 . The system reset to the initial payoff matrix (T=5 ,R=4 ,P= -2 . S= -3) every 500 rounds (standards interval = 500).(left) Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game.(right)Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game . Self-control was not achieved but there is a small increase on CC states compared to the baseline.*

The best performance was attained when we used smaller intervals for the standards. The overall average outcome that the system produced in CC states with the standards interval set to 125 and 75 was 46% and 50%, respectively (figures 4.4 p-q). It is noteworthy that the overall accumulated reward was also increased by around 3000 compared to the baseline (figure 4.4 r). When large intervals were used, the system did not manage to converge to a different outcome because of the large duration of the negative circumstances before resetting to the standards. However, the performance of the organisms was significantly improved when smaller interval was used.

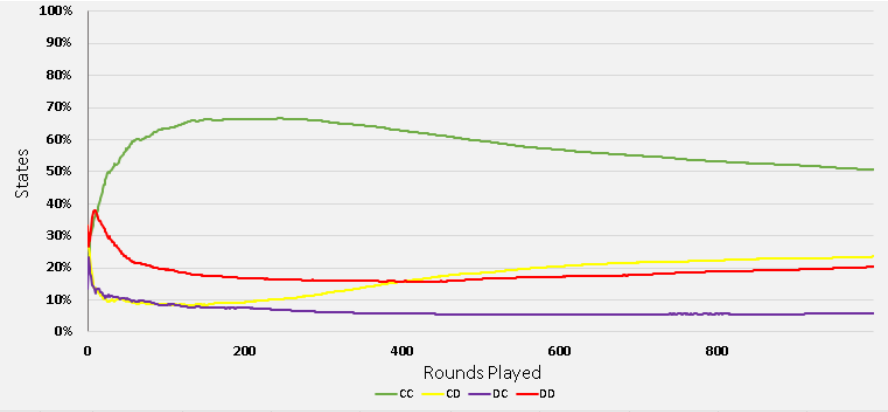

Figure 4.4 p: *Average of the outcomes CC, CD, DC and DD during 1000 rounds of the Q-learning agents playing the IPD game. Increasing T and decreasing R payoff. Ratio T=0.1,R=0.1 and payoff interval=25 . The system reset to the initial payoff matrix (T=5 ,R=4 ,P= -2 . S= -3) every 75 rounds (standards interval =75).*

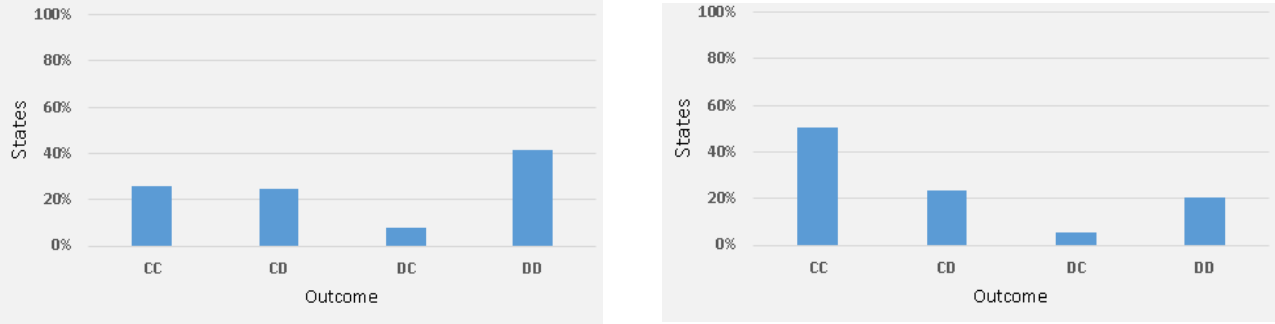

Figure 4.4 q: *Initial payoff matrix (T=5, R=4 , P=-2 , S=-3) . Increasing the T and decreasing R payoff. Ratio T=0.1 and payoff interval=25. (left) Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game (baseline) .(right) Overall average outcomes after 1000 rounds of the Q-learning agents playing the IPD game with standards interval = 75.*

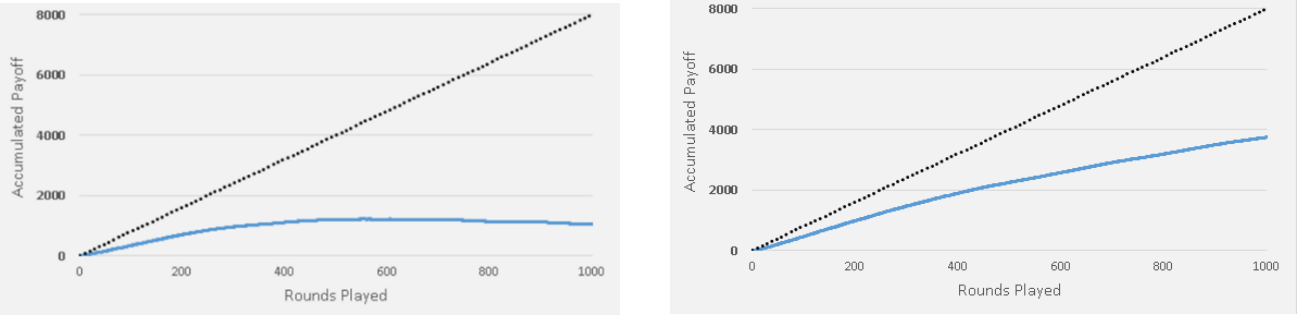

Figure 4.4 r: *Overall performance of the Q-learning agents during the 1000 rounds of the IPD game (blue line). The theoretically best performance is shown for comparison (dot-dashed line). Initial Payoff Matrix (T=5, R=4 , P=-2 , S=3). Increasing the T and decreasing R payoff. Ratio T=0.1,R=0.1 and payoff interval=25 . (left) Baseline results. (right) Standards interval set to 75. Self-control was achieved with an enormous increase on the accumulated payoff reward (3000 more).*

4.4.3 Summary and discussion on self-control virtue theory

In section 4.4 we presented the results of the simulations related to the self-control virtue theory. Self-control virtue theory suggests that self-regulation can be analyzed into three main ingredients: standards, monitoring, and operations that alter the self (Baumeister & Exline ,2001).

We deployed the presence of negative emotions in our model by changing the T and R payoff values periodically. At first, we decreased the R payoff value alone with ratio 0.1 and we tested different intervals on how often the matrix is reset to the initial values (the standards interval). With large standards intervals the improvement on self-control was quite satisfying but the best outcomes were attained when smaller intervals were used (100 or 125). Similar results were found when only T payoff value increased alone with the same ratio. Finally, we increased the intensity of the negative emotion by simultaneously reducing and increasing R and T, respectively. This time large intervals for the standards did not manage to influence quite enough the overall outcome. The intensity of the negative emotion was only successfully countered when small intervals for the standards were used. Obviously the indented self-control outcome was not always attained, however when good intervals are used this method amplitudes self-control and the overall accumulated payoff. Our findings seem to support most of our initial hypothesis for the self-regulation virtue theory.

Chapter 5

Conclusions and Future Work

5.1 Overview and conclusions ……..63
5.2 Future Work ……66

5.1 Overview and conclusions

The current thesis examined the influence of self-control theories when they are incorporated on a cognitive computational model. The theories are aligned with the idea that self-control should be viewed as a skill, a virtue and as a muscle, and thus should be treated accordingly. The computational model was inspired from a cognitive neuroscience model suggested by Rachlin (2000). He defined self-control as a dilemma between a smaller sooner reward (SS) and a large later reward (LL), that is experienced as an internal conflict between the higher part of the brain that is responsible for cognition and the lower part of the brain that is responsible for motivation. To express the self-control dilemma and the interaction between the two parts of the brain, we deployed a general sum game, the Prisoner's Dilemma game. Two agents representing the upper and lower parts of the brain engage in a variation of the PD game, the Iterated Prisoner's Dilemma (IPD) game, with the goal of learning to cooperate and thus achieve self-control. The Q-learning algorithm is used for training and the values of the payoff matrix in the IPD game provide the reinforcement signals that the agents receive according to their collective chosen actions during each round of the game.

The first concept we examined in the thesis is the self-control "strength model", introduced by Muraven and Baumeister (2000). The model entails that self-control operates like a muscle and the capability of exhibiting self-control relies on a limited resource, or self-control strength, which

depletes after any self-control operation. Initially, we were interested in addressing how self-control fatigue is experienced in our model. We observed that at the beginning of the IPD game the system always converges towards the self-control outcome (CC states), it then peaks at a certain round and finally there is a downward trend with self-control constantly decreasing. Although at first the system was able to distinguish the most beneficial collective outcome, when self-control resources were essentially depleted, exhibiting self-control became much harder. This phenomenon is caused by the lack of self-control resources and in psychology is referred as the ego depletion period (Muraven&Slessareva,2003; Muraven&Baumeister,2000).

Moving on, through our simulations we have showed that by increasing the intensity of internal conflict that is experienced during a self-control task, more self-control resources are required during each round of the game which causes the ego depletion period to arrive much earlier. In other words, this indicates that tasks which require more self-control are more affected by depletion than tasks which require less self-control (Muraven and Baumeister ,2000). The intensity of internal conflict was expressed and implemented in the model based on Cleanthous (2010) previous work.

The next concept, we incorporated was the self-control muscle recovery. The idea here was to alternate the system between the initial and an easier self-control task, to allow for recovery to take place. By this means we achieved to delay the ego depletion period and improved the overall performance. Afterwards, the concept of self-control task switching was further developed with the goal of exploring the overall performance when an initial task is followed by subsequent tasks of similar or different response conflict. When multiple subsequent tasks of similar conflict were presented during the game, we documented a significant increase on self-control performance. On the other hand, we observed a decreased performance when the initial task was followed by a different response conflict task. Our findings are aligned with previous studies done by Dewitte, Bruyneel and Geyskens (2009).

The next theory we considered, was extrinsic motivation which refers to the behavior that is driven by external factor such as a reward or a negative outcome (Berlyne, 1966). At first, we deployed the appearance of an external reward during the IPD game by increasing the lower brain agent's

motivation towards the large later (LL) reward. The round in which the external reward was presented in the system defined how much self-control was improved. Elimination of the ego depletion period was found when the external reward arrived at a round near the documented maximum outcome in CC states. Furthermore, we deployed an external authority that motivates the two agents towards the self-control behavior by forcing them periodically to cooperate. When the interval of the forced rounds was large the overall performance remained the same. However, with smaller intervals the influence of the external motivator was crucial, and the agents managed to sustain the strategy of cooperating throughout the game. We documented the higher outcome in CC states with this method, around 80%.

The final concept, we examined in this thesis is the self-control virtue theory. The theory suggests that self-regulation can be analyzed into three main ingredients: standards, monitoring, and operations that alter the self (Baumeister & Exline ,2001). Our goal was to discover how could these three ingredients be applied while the system is experiencing the effects of the presence of negative emotions. We deployed the negative emotions based on Nikodemou (2020) and we simulated the operations that recorrect the system towards its relevant standards. When the experienced emotion was of moderate intensity, self-control was significantly improved despite how often the virtue operations was applied. However, with a high intensity emotion, it was required to apply the operations much more often for the system to manage to sustain the desired behavior despite the negative circumstances.

In conclusion, we achieved to incorporate many theories that affect positively or negatively self-control in a computational model. The analyzed concepts point out on the belief that self-control should be viewed and treated as a skill, a muscle, and a virtue. Each implemented theory has a psychological background and is supported by findings of neuroscientific research; therefore the model is cognitively adequate. Furthermore, the low complexity of the model in combination with the nature of the PD game allowed us to express each self-control concept in a precise manner. Self-control provides the power to do what is right and is considered to be one of the most important indicators of life success. Therefore, challenging ourselves daily and practicing self-control should be our primary ambition.

5.2 Future Work

We investigated many different concepts, some were highly related to one another and others were not. Perhaps, the most important theory was the strength model of self-control that provides one of the most influential explanation according to psychologists, on what causes self-control failure (Muraven and Baumeister ,2000). All the implemented theories have been individually examined in order to be unambiguous of what caused the corresponding change in self-control every time. For that reason, one by one the theories could stand alone and provide the bases for further research. Moreover, the next step would be to build a more complex self-control scenario in which multiple concepts are been combined to express the behavior in a more collective manner. We already did the first step, by combining the effects of emotions provided by Nikodemou (2020) and intensity of the internal conflict provided by Cleanthous (2010), however there is still potential for much more development.

The definition of self-control allowed us to design a model of two agents representing the lower and higher parts of the brain. One agent is driven towards smaller sooner (SS) rewards while the other is driven towards large later (LL) rewards. The agents are experiencing a dilemma and their goal is to learn the best strategy to maximize their reward, which is to cooperate in our case. In the business world, companies are dealing with similar dilemmas where a short-term investment could appear rewarding however a more patient and more methodical long-term approach can bring a much higher benefit. Therefore, the concepts we deployed in the computational model of self-control could serve as an implementation in a more practical model of autonomous agents that are driven by short-term and long-term motives, and ought to collectively decide the most beneficial and balanced option to maximize a company's profit.

Finally, throughout human evolution it is not clear why self-control was not perfectly evolved. The most efficient strategy that was developed by humans to solve self-control problems is the precommitment behavior. Precommitment is defined as making a choice now with the specific goal of denying (or at least restricting) oneself future choices (Rachlin, 1995). This self-control strategy however is not the optimal because it presupposes that the current choice would remain the best option in the future. Something that is not always correct, especially in the unpredictable

world we live in. Christodoulou, Banfield, & Cleanthous (2010) previously proven that the effects of precommitment behavior could be model by applying a differential bias (Ψ) on the accumulated payoff of CD and DC states . A negative differential bias (Ψ) is applied to the CD state in order to decrease the motivation of the system towards the smaller sooner (SS) reward and at the same time a positive differential bias (Ψ) is applied to the DC state in order to increase the motivation of the system towards the large later (LL) reward . For that reason, as we increase the differential bias Ψ, the incentive of the system to cooperate is promoted. Implementing genetic algorithms in our computational model could give us an insight on why the evolved strategy of humans for solving self-control problems is precommitment.

One possible approach to be taken here is to have a population of pairs of lower and higher brain agents. Each pair has a different IPD payoff matrix at the beginning and in each generation the pairs of the population are playing the IPD game for 1000 rounds. The pairs that achieve the highest CC outcome at the end, have a larger probability to be selected as the parents of the next generation. A crossover operation follows on the IPD payoff matrixes of the selected parents and a new population of pairs is generated with evolved IPD payoff matrixes. An initial solution for the crossover operation could be to change the accumulated payoff for each state of IPD matrix of the generated child pair based on the averages of the accumulated payoff of the corresponding states that are calculated by the IPD payoff matrixes of the two parents. Independently of the crossover operation that is chosen, a mechanism should guarantee that the two rules of the IPD game are not violated throughout the evolution. Finally, as we proceed into the next generations the pairs of agents that "survive" are those that developed the best strategy for solving the problem of self-control. If the precommitment behavior was indeed the evolved self-control strategy of the final population, then we must observe an increased and decreased accumulated payoff of the DC and CD states respectively, compared to the initial accumulated payoffs of the corresponding states in the IPD matrix of each pair.

References

Banfield, G. D. (2006). Simulation of Self-Control through Precommitment Behaviour in an Evolutionary System. Ph.D., Birkbeck, University of London.

Baumeister, R. F., Heatherton, T. F., & Tice, D. M. (1994). Losing control: How and why people fail at self-regulation. San Diego, CA: Academic Press

Baumeister, R. F., & Heatherton, T. F. (1996). Self-regulation failure: An overview. Psychological Inquiry, 7, 1–15.

Baumeister, R. F. (1997). Evil: Inside human violence and cruelty. New York: W.H. Freeman.

Baumeister .R.F. & Exline J.J. (2001). Virtue, Personality, and Social Relations: Self-Control as the Moral Muscle. Journal of Personality, 67(6), 1165–1194.

Baumeister, R. F., Gailliot, M., DeWall, C. N., & Oaten, M. (2006). Self-regulation and personality: How interventions increase regulatory success, and how depletion moderates the effects of traits on behavior. Journal of Personality, 74, 1773-1802.

Berlyne D. E.(1966). Curiosity and exploration. Science,153, 25-33

Bruyneel , Dewitte, Vohs and Warlop (2006). "Repeated Choosing Increases Susceptibility to Affective Product Features," International Journal of Research in Marketing, 23 (2), 215-225.

Carver, C. S., & Scheier, M. F. (1981). Attention and self-regulation: A control-theory approach to human behavior. New York,NY: Springer-Verlag.

Christodoulou, C., Banfield, G., & Cleanthous, A. (2010). Self-control with spiking and non-spiking neural networks playing games. Journal of Physiology, Paris, 104(3–4), 108–117.

Cleanthous, A. (2010). In search of Self-control through computational modelling of Internal Conflict. PhD Thesis, University of Cyprus .

Cyders & Smith, 2008; Heatherton, 2011; Schmeichel & Tang, 2015Ruderman, A. J. (1985). Dysphoric mood and overeating: A test of restraint theory's disinhibition hypothesis. Journal of Abnormal Psychology, 94(1), 78–85.

Dewitte, S., Bruyneel, S., & Geyskens, K. (2009). Self-regulating enhances self-regulation in subsequent consumer decisions involving similar response conflicts. Journal of Consumer Research, 36(3), 394–405.

Duckworth, A. L., & Seligman, M. E. (2005). Self-discipline outdoes IQ in predicting academic performance of adolescents. Psychological science, 16(12), 939-944.

Gailliot, M.T. and Baumeister, R.F. (2007). The physiology of willpower: Linking blood glucose to self-control. Personality and Social Psychology Review**,** 11(4), 303–27.

Georgiou, A. (2015). Μελέτη σχέσης αυτοελέγχου και συνειδητότητας. Bachelor's thesis. University of Cyprus.

Hare, T. A., Camerer, C. F., & Rangel, A. (2009). Self-control in decision-making involves modulation of the vmPFC valuation system. Science , 324(5927), 646–648.

Hull, J. G. (1981). A self-awareness model of the causes and effects of alcohol consumption. Journal of Abnormal Psychology, 90, 586–600

Kavka, G. S. (1991). Is Individual Choice Less Problematic than Collective Choice? Economics & Philosophy, 7(2), 143–165

Kahneman, D. (2011). Thinking, fast and slow. New York: Farrar, Straus and Giroux

McClure, S. M., Laibson, D. I., Loewenstein, G., & Cohen, J. D. (2004). Separate neural systems value immediate and delayed monetary rewards. Science, 306(5695), 503–507.

Moffitt, T. E., Arseneault, L., Belsky, D., Dickson, N., Hancox, R. J.,Harrington, H., et al. (2011). A gradient of childhood self-control predicts health, wealth, and public safety. Proceedings of the National Academy of Sciences of the United States of America,108, 2693–2698.

Muraven, M. and Slessareva, E. (2003). Mechanisms of Self-Control Failure: Motivation and Limited Resources. Personality and Social Psychology Bulletin, 29, 894-906.

Muraven, M., Baumeister, R. F. and Tice, D. M. (1999). Longitudinal improvement of self-regulation through practice: building self-control strength through repeated exercise. Journal of Social Psychology, 139, 446-457.

Muraven, M. and Baumeister, R. F. (2000). Self-Regulation and deplation of a limited sources: Does self-control resemble a muscle? Psychological Bulletin, 126(2), 247-259.

Muraven, M., Tice, D. M. and Baumeister, R. F. (1998). Self-control as a limited resource: regulatory depletion patterns. Journal of Personality and Social Psychology, 74, 774-789.

Nash, J. F. (1950). Equilibrium Points in n-Person Games. Proceedings of the National Academy of Sciences of the United States of America, 36(1), 48–49.

Nikodemou, A. (2020). Positive and negative emotions in a computational model of self-control, BSc Thesis, Department of Computer Science, University of Cyprus.

Pinder, C.C. (1998). Work motivation in organizational behavior.Upper Saddle River, NJ: Prentice-Hall.

Rachlin, H. (2000). The Science of Self-Control. Harvard University Press, Cambridge, MA.

Rachlin, H. (1995). Self-control: Beyond commitment. Behavioral and Brain Sciences, 18(1), 109–121.

Rubinstein, A. (1982). Perfect Equilibrium in a Bargaining Model. Econometrica, 50(1), 97–109.

Ruderman, A. J. (1985). Dysphoric mood and overeating: A test of restraint theory's disinhibition hypothesis. Journal of Abnormal Psychology, 94(1), 78–85.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. Machine Learning, 3(1), 9–44.

Von Neumann, J. and Morgenstern, O. (1947). Theory of games and economic behavior. Princeton University Press, Princeton, NJ.

Wörgötter, F., & Porr, B. (2008). Reinforcement learning. Scholarpedia, 3, 1448.

Appendix A

The development of the basic Q-Learning model (Appendices A-B) is by Georgiou (2015).

Class MainQ.java

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Random;

/**
 * The basic model was created by Jeannette Chahwan & Anna Georgiou on 20/01/15.
   Later updated by Kyriakos Georgiou 2021.
 */
public class MainQ {
   public static Player lower;
   public static Player higher;
   int momentum=50;
   public static double[][][] states_results;
   public static double[][][] payoff_results;
   public static double[] overall_payoff;

   public static double[] overall_payoff_neutral;
   public static double[][][] payoff_results_neutral;

   public static double T, R, P, S;

   public static int get_state(int action_lower, int action_higher) {
      int next_state;

      if(action_lower==0 && action_higher==0)
         next_state = 0;
      else if(action_lower==0 && action_higher==1)
         next_state = 2;
      else if(action_lower==1 && action_higher==0)
         next_state = 1;
      else
         next_state = 3;

      return next_state;
   }

   public static void update_states_results(int episode, int state, int trial){
      states_results[episode][trial][0] = states_results[episode-1][trial][0];
      states_results[episode][trial][1] = states_results[episode-1][trial][1];
      states_results[episode][trial][2] = states_results[episode-1][trial][2];
      states_results[episode][trial][3] = states_results[episode-1][trial][3];
```

```
    states_results[episode][trial][state]++;
  }

  public static void update_payoff_results(int episode, double lower_payoff, double higher_payoff, int
trial){
    payoff_results[episode][trial][0] = payoff_results[episode-1][trial][0];
    payoff_results[episode][trial][1] = payoff_results[episode-1][trial][1];

    payoff_results[episode][trial][0] += lower_payoff;
    payoff_results[episode][trial][1] += higher_payoff;

    // for neutral emotion
    if ((lower_payoff == S && higher_payoff == T) || (lower_payoff == T && higher_payoff == S)) {
      payoff_results_neutral[episode][trial][0] = payoff_results_neutral[episode - 1][trial][0];
      payoff_results_neutral[episode][trial][1] = payoff_results_neutral[episode - 1][trial][1];

    }
    else {
      payoff_results_neutral[episode][trial][0] = payoff_results_neutral[episode - 1][trial][0];
      payoff_results_neutral[episode][trial][0] += lower_payoff;
      payoff_results_neutral[episode][trial][1] = payoff_results_neutral[episode - 1][trial][1];
      payoff_results_neutral[episode][trial][1] += higher_payoff;
    }
  }

  public static void statistics(int numberOfTrials, int numberOfEpisodes){
    int e, t, s, p;
    double sumAll_states;
    double sumAll_payoffs;

    for(e=0; e<numberOfEpisodes; e++){
      for(t=0; t<numberOfTrials; t++){   // find sum of states visits from each trial of current episode and
save in last column
        for(s=0; s<4; s++){    // states
          states_results[e][numberOfTrials][s] += states_results[e][t][s];
        }

        for(p=0; p<2; p++){    // payoffs
          payoff_results[e][numberOfTrials][p] += payoff_results[e][t][p];
          payoff_results_neutral[e][numberOfTrials][p] += payoff_results_neutral[e][t][p];
        }
      }
      sumAll_states=0;
      sumAll_payoffs=0;

      for(s=0; s<4; s++){    // sum of state fields in last column (#trials+1)
        sumAll_states += states_results[e][numberOfTrials][s];
      }

      for(s=0; s<4; s++){    // normalise
        states_results[e][numberOfTrials][s] /= sumAll_states;
      }

      for(p=0; p<2; p++){    // sum of payoff fields in last column (#trials+1)
        sumAll_payoffs += payoff_results[e][numberOfTrials][p];
```

```
      }

      overall_payoff[e] = sumAll_payoffs/numberOfTrials;
   }

}

//Use this function if you want to print the state and payoff results
public static void print_statistics(int numberOfTrials, int numberOfEpisodes){
   int e, s, p;

   // states
   for(e=0; e<numberOfEpisodes; e++){
      for(s=0; s<4; s++){
         System.out.printf("%.4f ", states_results[e][numberOfTrials][s]);
      }
      System.out.println();
   }

   // payoffs
   for(e=0; e<numberOfEpisodes; e++){
      for(p=0; p<2; p++){
         System.out.printf("%.4f ", payoff_results[e][numberOfTrials][p]);
      }
      System.out.println();
   }
}

public static void printToFile_statistics(int numberOfTrials, int numberOfEpisodes){
   int e, s, p;

   // write results to file
   try {
      File file_s = new File("states_results.txt");
      File file_p = new File("payoff_results.txt");

      // if file doesnt exists, then create it
      if (!file_s.exists()) {
         file_s.createNewFile();
      }

      if (!file_p.exists()) {
         file_p.createNewFile();
      }

      FileWriter fw_s = new FileWriter(file_s.getAbsoluteFile());
      FileWriter fw_p = new FileWriter(file_p.getAbsoluteFile());
      BufferedWriter bw_s = new BufferedWriter(fw_s);
      BufferedWriter bw_p = new BufferedWriter(fw_p);
      bw_s.write("Rounds\tCC\tCD\tDC\tDD\n");
      for(e=0; e<numberOfEpisodes; e++){
         bw_s.write((e+1)+"\t");
         for(s=0; s<4; s++){
            bw_s.write(Double.toString(states_results[e][numberOfTrials][s]));
            bw_s.write("\t");
```

```
            }
            bw_s.write("\n");

            for(p=0; p<2; p++){
               bw_p.write(Double.toString(payoff_results[e][numberOfTrials][p]/numberOfTrials));
               bw_p.write("\t");
            }
            bw_p.write(Double.toString(overall_payoff[e]));
            bw_p.write("\t");
            bw_p.write(Double.toString(overall_payoff_neutral[e]));
            bw_p.write("\n");
         }

         bw_s.close();
         bw_p.close();
      } catch (IOException ex) {
         ex.printStackTrace();
      }
   }
   /*
      This is the main function of the Program. The variable option defines which theory - function is going
to be
      tested each time .
    */
   public static void main(String[] args) {
      // The option variable
      int option =1;
      switch(option) {
         case 1:
            System.out.println("Chosen method : Round Depletion");
            Round_Depletetion(1000);
            break;
         case 2:
            System.out.println("Chosen method : Muscle Recovery");
            Muscle_Recovery(15,30);
            break;
         case 3:
            System.out.println("Chosen method : Task Switching ");
            TaskSwitching(true,125);
            break;
         case 4:
            System.out.println("Chosen method : External Monetary Reward ");
            External_Monetary_Reward(300);
            break;
         case 5:
            System.out.println("Chosen method : External Motivator");
            External_Motivator(20);
            break;
         case 6:
            System.out.println("Chosen method : Clear Standards ");
            Clear_Standards(25,100 , 0 ,0.1);
            break;
         default:
            // Wrong Option - Give an option from 1-6
            System.out.println("Please Set the option to be from 1-6");       }
```

```
  }

  /**
   *  This is the basic method for the Self-control model
   *  The caller provides only the number of episodes (rounds) that the IPD game lasts
   **/
  public static void  Round_Depletetion(int episodes){
    int initial_state, current_state, next_state; // Takes values 0-3
    int i;
    int episodes_before = episodes;
    int episodes_after = 000;
    int trials = 15;
    states_results = new double[episodes_before + episodes_after][trials+1][4];
    payoff_results = new double[episodes_before + episodes_after][trials+1][2];
    overall_payoff = new double[episodes_before + episodes_after];

    payoff_results_neutral = new double[episodes_before + episodes_after][trials+1][2];
    overall_payoff_neutral = new double[episodes_before + episodes_after];

    T=5; R=4; P=-2; S=-3;

    for(int t=0; t<trials; t++) {
      lower = new Player();
      lower.setDiscount(0.9);
      lower.setLearning_rate(0.1);
      lower.setEpsilon(0.1);
      lower.setPayoff_matrix(4, 7.5, -4.5, -2);

      higher = new Player();
      higher.setDiscount(0.1);
      higher.setLearning_rate(0.1);
      higher.setEpsilon(0.1);
      higher.setPayoff_matrix(4, -4.5, 7.5, -2);

      initial_state = (int) (Math.random() * 4);
      current_state = initial_state;
      states_results[0][t][current_state]++;

      for (i = 1; i < episodes_before; i++) {
        lower.setCurrent_action(lower.choose_action(current_state));
        higher.setCurrent_action(higher.choose_action(current_state));


        next_state = get_state(lower.getCurrent_action(), higher.getCurrent_action());

        lower.update_Q(current_state, next_state);
        higher.update_Q(current_state, next_state);
        current_state = next_state;

        update_states_results(i, current_state, t);
        update_payoff_results(i, lower.getPayoff_matrix(current_state),
higher.getPayoff_matrix(current_state), t);
      }

      // Set epsilon = 0 and initial payoff matrix
```

```
        lower.setEpsilon(0);
        higher.setEpsilon(0);
        R=4; T=5; S=-3; P=-2;
        lower.setPayoff_matrix(4, 5, -3, -2);
        higher.setPayoff_matrix(4, -3, 5, -2);

        for (i = 1; i <= episodes_after; i++) {
          lower.setCurrent_action(lower.choose_action(current_state));
          higher.setCurrent_action(higher.choose_action(current_state));

          next_state = get_state(lower.getCurrent_action(), higher.getCurrent_action());

          lower.update_Q(current_state, next_state);
          higher.update_Q(current_state, next_state);
          current_state = next_state;

          update_states_results(episodes_before+i-1, current_state, t);
          update_payoff_results(episodes_before+i-1, lower.getPayoff_matrix(current_state),
higher.getPayoff_matrix(current_state), t);
        }
      }

      statistics(trials, (episodes_before+episodes_after));
      printToFile_statistics(trials, (episodes_before+episodes_after));

    }

    /**
     *  The function is used to implement the Muscle recovery concept
     *  recover var. defines how many rounds the agents are staying in the recovery state
     *  maxDays var. defines the maximum continuous Rounds that the agent interact on the normal payoff
matrix before entering recovery
     */
    public static void Muscle_Recovery(int recover , int maxDays){
      if(recover > maxDays)
      {
        System.out.println("Wrong values !\n Recover days cannot exceed maxDays (max continuous Days
working out the Self control muscle)");
        return;
      }
      int initial_state, current_state, next_state; // Takes values 0-3
      int i;
      int episodes_before = 1000, episodes_after = 000;
      int trials = 15;
      states_results = new double[episodes_before + episodes_after][trials+1][4];
      payoff_results = new double[episodes_before + episodes_after][trials+1][2];
      overall_payoff = new double[episodes_before + episodes_after];

      payoff_results_neutral = new double[episodes_before + episodes_after][trials+1][2];
      overall_payoff_neutral = new double[episodes_before + episodes_after];

      T=5; R=4; P=-2; S=-3;

      for(int t=0; t<trials; t++) {
        lower = new Player();
```

```
        lower.setDiscount(0.9);
        lower.setLearning_rate(0.1);
        lower.setEpsilon(0.1);
        lower.setPayoff_matrix(4, 5, -3, -2);

        higher = new Player();
        higher.setDiscount(0.1);
        higher.setLearning_rate(0.1);
        higher.setEpsilon(0.1);
        higher.setPayoff_matrix(4, -3, 5, -2);

        initial_state = (int) (Math.random() * 4);
        current_state = initial_state;
        states_results[0][t][current_state]++;

        int counter =0;
        int rest_counter=0;
        int restFlag=0;
        for (i = 1; i < episodes_before; i++) {
           counter++;
           lower.setCurrent_action(lower.choose_action(current_state));
           higher.setCurrent_action(higher.choose_action(current_state));


           next_state = get_state(lower.getCurrent_action(), higher.getCurrent_action());

           lower.update_Q(current_state, next_state);
           higher.update_Q(current_state, next_state);

           current_state = next_state;

           update_states_results(i, current_state, t);
           update_payoff_results(i, lower.getPayoff_matrix(current_state),
higher.getPayoff_matrix(current_state), t);

           if(counter>=maxDays){
              // Enter Recovery State
              lower.setPayoff_matrix(4, 4.5, -2.5, -2);
              higher.setPayoff_matrix(4, -2.5,4.5, -2);

              rest_counter=recover;
              counter=0;
              restFlag=1;
              continue;
           }
           // Rest - counter defines the number of rounds left before entering the next recovery state
           else if(rest_counter>0 && restFlag==1){
              rest_counter--;
           }

           else if(rest_counter==0 && restFlag==1) {
              // Back to the Initial Matrix  - Recovery
              restFlag=0;
              lower.setPayoff_matrix(4, 5, -3, -2);
              higher.setPayoff_matrix(4, -3, 5, -2);
```

```
            counter=0;
          }

        }

        // Set epsilon = 0 and initial payoff matrix
        lower.setEpsilon(0);
        higher.setEpsilon(0);
        R=4; T=5; S=-3; P=-2;
        lower.setPayoff_matrix(4, 5, -3, -2);
        higher.setPayoff_matrix(4, -3, 5, -2);

        for (i = 1; i <= episodes_after; i++) {
          lower.setCurrent_action(lower.choose_action(current_state));
          higher.setCurrent_action(higher.choose_action(current_state));

          next_state = get_state(lower.getCurrent_action(), higher.getCurrent_action());

          lower.update_Q(current_state, next_state);
          higher.update_Q(current_state, next_state);

          current_state = next_state;

          update_states_results(episodes_before+i-1, current_state, t);
          update_payoff_results(episodes_before+i-1, lower.getPayoff_matrix(current_state),
higher.getPayoff_matrix(current_state), t);

          if(counter>=maxDays){
            // Enter Recovery State
            lower.setPayoff_matrix(4, 4.5, -2.5, -2);
            higher.setPayoff_matrix(4, -2.5,4.5, -2);

            rest_counter=recover;
            counter=0;
            restFlag=1;
            continue;
          }
          // Rest - counter defines the number of rounds left before entering the next recovery state
          else if(rest_counter>0 && restFlag==1){
            rest_counter--;
          }

          else if(rest_counter==0 && restFlag==1) {
            // Back to the Initial Matrix  - Recovery
            restFlag=0;
            lower.setPayoff_matrix(4, 5, -3, -2);
            higher.setPayoff_matrix(4, -3, 5, -2);
            counter=0;
          }
        }
      }

      statistics(trials, (episodes_before+episodes_after));
      printToFile_statistics(trials, (episodes_before+episodes_after));
```

```
}

/**
 *  Task switching to Similar Conflict Task - (when true) or Different Conflict Task (when false)
 *  rounds variable - indicates how often swapping to new task happens
 */
public static void TaskSwitching(boolean Similar_Conflict ,int rounds){

   int initial_state, current_state, next_state; // Takes values 0-3
   int i;
   int episodes_before = 1000, episodes_after = 000;
   int trials = 15;
   states_results = new double[episodes_before + episodes_after][trials+1][4];
   payoff_results = new double[episodes_before + episodes_after][trials+1][2];
   overall_payoff = new double[episodes_before + episodes_after];

   payoff_results_neutral = new double[episodes_before + episodes_after][trials+1][2];
   overall_payoff_neutral = new double[episodes_before + episodes_after];

   T=5; R=4; P=-2; S=-3;

   for(int t=0; t<trials; t++) {
      lower = new Player();
      lower.setDiscount(0.9);
      lower.setLearning_rate(0.1);
      lower.setEpsilon(0.1);
      lower.setPayoff_matrix(4, 5, -3, -2);

      higher = new Player();
      higher.setDiscount(0.1);
      higher.setLearning_rate(0.1);
      higher.setEpsilon(0.1);
      higher.setPayoff_matrix(4, -3, 5, -2);

      initial_state = (int) (Math.random() * 4);
      current_state = initial_state;
      states_results[0][t][current_state]++;

      for (i = 1; i < episodes_before; i++) {

         lower.setCurrent_action(lower.choose_action(current_state));
         higher.setCurrent_action(higher.choose_action(current_state));

         next_state = get_state(lower.getCurrent_action(), higher.getCurrent_action());

         lower.update_Q(current_state, next_state);
         higher.update_Q(current_state, next_state);

         // Switch to a different task
         if ( i%rounds==0 ) {
            double RealConfilct = T-S;
            Random r = new Random();
            if (Similar_Conflict) {
               // Add Random values from  1 to 3
               double randomValue = r.nextInt(3)+1;
```

```
            // Similar Conflict - Add a Constant value to each payoff reward.
            T=T+randomValue;
            R =  R+randomValue;
            P =  P+randomValue;
            S =  S+randomValue;
          }
          else {
            while (true) {

              // Random values from 0 to 7
              T =  r.nextInt(8);
              // Random value from 0 to -7
              S = -r.nextInt(8);

              // Make sure the rules of IPD game are not violated
              if (RealConfilct != T - S &&
                   T > R && R > P && P > S && 2 * R > T + S) {
                // Good values
                break;
              }
            }
          }
          higher.setPayoff_matrix(R, S, T, P);
          lower.setPayoff_matrix(R, T, S, P);

          System.out.printf("R:%.4f T:%.4f S:%.4f P:%.4f e:%d t:%d\n", higher.getPayoff_matrix(0),
              higher.getPayoff_matrix(2), higher.getPayoff_matrix(1),
              higher.getPayoff_matrix(3), i, t);
        }

        current_state = next_state;
        update_states_results(i, current_state, t);
        update_payoff_results(i, lower.getPayoff_matrix(current_state),
higher.getPayoff_matrix(current_state), t);
      }

      // Set epsilon = 0 and initial payoff matrix
      lower.setEpsilon(0);
      higher.setEpsilon(0);
      R=4; T=5; S=-3; P=-2;
      lower.setPayoff_matrix(4, 5, -3, -2);
      higher.setPayoff_matrix(4, -3, 5, -2);


      for (i = 1; i <= episodes_after; i++) {
        lower.setCurrent_action(lower.choose_action(current_state));
        higher.setCurrent_action(higher.choose_action(current_state));

        next_state = get_state(lower.getCurrent_action(), higher.getCurrent_action());

        lower.update_Q(current_state, next_state);
        higher.update_Q(current_state, next_state);
        // Switch to a different task
        if ( i%rounds==0 ) {
```

```
            double RealConfilct = T-S;
            Random r = new Random();
            if (Similar_Conflict) {
               // Add Random values from  1 to 3
               double randomValue = r.nextInt(3)+1;

               // Similar Conflict - Add a Constant value to each payoff reward.
               T=T+randomValue;
               R =  R+randomValue;
               P =  P+randomValue;
               S =  S+randomValue;
            }
            else {
               while (true) {

                  // Random values from 0 to 7
                  T =  r.nextInt(8);
                  // Random value from 0 to -7
                  S = -r.nextInt(8);

                  // Make sure the rules of IPD game are not violated
                  if (RealConfilct != T - S &&
                        T > R && R > P && P > S && 2 * R > T + S) {
                     // Good values
                     break;
                  }
               }
            }
            higher.setPayoff_matrix(R, S, T, P);
            lower.setPayoff_matrix(R, T, S, P);

            System.out.printf("R:%.4f T:%.4f S:%.4f P:%.4f e:%d t:%d\n", higher.getPayoff_matrix(0),
                  higher.getPayoff_matrix(2), higher.getPayoff_matrix(1),
                  higher.getPayoff_matrix(3), i, t);
         }

         current_state = next_state;
         update_states_results(episodes_before+i-1, current_state, t);
         update_payoff_results(episodes_before+i-1, lower.getPayoff_matrix(current_state),
higher.getPayoff_matrix(current_state), t);
      }
    }
    statistics(trials, (episodes_before+episodes_after));
    printToFile_statistics(trials, (episodes_before+episodes_after));
  }



  /**
   * The function is used to implement an external reward that appears during the IPD game.
   * When the system reaches round discount_interval , the higher brain agent discount rate is reduce by
0.3
   */
  public static void External_Monetary_Reward(int discount_interval){
```

```
    int initial_state, current_state, next_state; // Takes values 0-3
    int i;
    int episodes_before = 1000, episodes_after = 000;
    int trials = 15;
    states_results = new double[episodes_before + episodes_after][trials+1][4];
    payoff_results = new double[episodes_before + episodes_after][trials+1][2];
    overall_payoff = new double[episodes_before + episodes_after];

    payoff_results_neutral = new double[episodes_before + episodes_after][trials+1][2];
    overall_payoff_neutral = new double[episodes_before + episodes_after];

    T=5; R=4; P=-2; S=-3;

    double discountRate= 0.3 ;  // Discount decrease factor

    for(int t=0; t<trials; t++) {
       lower = new Player();
       lower.setDiscount(0.9);
       lower.setLearning_rate(0.1);
       lower.setEpsilon(0.1);
       lower.setPayoff_matrix(4, 5, -3, -2);

       higher = new Player();
       higher.setDiscount(0.1);
       higher.setLearning_rate(0.1);
       higher.setEpsilon(0.1);
       higher.setPayoff_matrix(4, -3, 5, -2);

       initial_state = (int) (Math.random() * 4);
       current_state = initial_state;
       states_results[0][t][current_state]++;

       for (i = 1; i < episodes_before; i++) {

          lower.setCurrent_action(lower.choose_action(current_state));
          higher.setCurrent_action(higher.choose_action(current_state));


          next_state = get_state(lower.getCurrent_action(), higher.getCurrent_action());

          lower.update_Q(current_state, next_state);
          higher.update_Q(current_state, next_state);

          if ( i== discount_interval ){
             lower.setDiscount(lower.getDiscount()-discountRate);

          }

          current_state = next_state;

          update_states_results(i, current_state, t);
          update_payoff_results(i, lower.getPayoff_matrix(current_state),
higher.getPayoff_matrix(current_state), t);
       }
```

```
        // Set epsilon = 0 and initial payoff matrix
        lower.setEpsilon(0);
        higher.setEpsilon(0);
        // Set lower brain discount rate to the initial one
        lower.setDiscount(0.9);
        R=4; T=5; S=-3; P=-2;
        lower.setPayoff_matrix(4, 5, -3, -2);
        higher.setPayoff_matrix(4, -3, 5, -2);

        for (i = 1; i <= episodes_after; i++) {
           lower.setCurrent_action(lower.choose_action(current_state));
           higher.setCurrent_action(higher.choose_action(current_state));

           next_state = get_state(lower.getCurrent_action(), higher.getCurrent_action());

           lower.update_Q(current_state, next_state);
           higher.update_Q(current_state, next_state);

           if ( i== discount_interval ){
              lower.setDiscount(lower.getDiscount()-discountRate);

           }

           current_state = next_state;

           update_states_results(episodes_before+i-1, current_state, t);
           update_payoff_results(episodes_before+i-1, lower.getPayoff_matrix(current_state),
higher.getPayoff_matrix(current_state), t);
        }
     }

     statistics(trials, (episodes_before+episodes_after));
     printToFile_statistics(trials, (episodes_before+episodes_after));

   }

   /**
    *  The function is used to implement an external motivator ( authority) that forces the agents to
cooperate.
    *  When the system reaches a round that is divided exactly by the forced_rounds_interval the agents are
forced to choose their next actions
    *  to be cooperation .
    */
   public static void External_Motivator(int forced_rounds_interval){

     int initial_state, current_state, next_state; // Takes values 0-3
     int i;
     int episodes_before = 1000, episodes_after = 000;
     int trials = 15;
     states_results = new double[episodes_before + episodes_after][trials+1][4];
     payoff_results = new double[episodes_before + episodes_after][trials+1][2];
     overall_payoff = new double[episodes_before + episodes_after];

     payoff_results_neutral = new double[episodes_before + episodes_after][trials+1][2];
```

```
    overall_payoff_neutral = new double[episodes_before + episodes_after];

    T=5; R=4; P=-2; S=-3;



    for(int t=0; t<trials; t++) {
       lower = new Player();
       lower.setDiscount(0.9);
       lower.setLearning_rate(0.1);
       lower.setEpsilon(0.1);
       lower.setPayoff_matrix(4, 5, -3, -2);

       higher = new Player();
       higher.setDiscount(0.1);
       higher.setLearning_rate(0.1);
       higher.setEpsilon(0.1);
       higher.setPayoff_matrix(4, -3, 5, -2);

       initial_state = (int) (Math.random() * 4);
       current_state = initial_state;
       states_results[0][t][current_state]++;

       for (i = 1; i < episodes_before; i++) {

          // External Authority Forcing the agents to Both Cooperate
          if ( i% forced_rounds_interval ==0){
             lower.setCurrent_action(0);
             higher.setCurrent_action(0);
          }
          else{
             lower.setCurrent_action(lower.choose_action(current_state));
             higher.setCurrent_action(higher.choose_action(current_state));
          }


          next_state = get_state(lower.getCurrent_action(), higher.getCurrent_action());

          lower.update_Q(current_state, next_state);
          higher.update_Q(current_state, next_state);

          current_state = next_state;

          update_states_results(i, current_state, t);
          update_payoff_results(i, lower.getPayoff_matrix(current_state),
higher.getPayoff_matrix(current_state), t);
       }

       // Set epsilon = 0 and initial payoff matrix
       lower.setEpsilon(0);
       higher.setEpsilon(0);
       R=4; T=5; S=-3; P=-2;
       lower.setPayoff_matrix(4, 5, -3, -2);
       higher.setPayoff_matrix(4, -3, 5, -2);
```

```
        for (i = 1; i <= episodes_after; i++) {
          lower.setCurrent_action(lower.choose_action(current_state));
          higher.setCurrent_action(higher.choose_action(current_state));

          next_state = get_state(lower.getCurrent_action(), higher.getCurrent_action());

          lower.update_Q(current_state, next_state);
          higher.update_Q(current_state, next_state);

          current_state = next_state;

          update_states_results(episodes_before+i-1, current_state, t);
          update_payoff_results(episodes_before+i-1, lower.getPayoff_matrix(current_state),
higher.getPayoff_matrix(current_state), t);
        }
      }

      statistics(trials, (episodes_before+episodes_after));
      printToFile_statistics(trials, (episodes_before+episodes_after));

    }

    /**
     *  This function is used to model Self-control as a virtue.
     *  The agents interact in the IPD game while experiencing changes in R and T values.
     *  Those changes collapse each time payoff_interval rounds pass and the matrix is set to default after
     *  standards_inverval number of rounds.
     */
    public static void Clear_Standards(int payoff_interval,int standards_interval , double ratio_R ,double
ratio_T){

      int initial_state, current_state, next_state; // Takes values 0-3
      int i;
      int episodes_before = 1000, episodes_after = 000;
      int trials = 15;
      states_results = new double[episodes_before + episodes_after][trials+1][4];
      payoff_results = new double[episodes_before + episodes_after][trials+1][2];
      overall_payoff = new double[episodes_before + episodes_after];

      payoff_results_neutral = new double[episodes_before + episodes_after][trials+1][2];
      overall_payoff_neutral = new double[episodes_before + episodes_after];

      T=5; R=4; P=-2; S=-3;

      int flag=0;

      for(int t=0; t<trials; t++) {
        lower = new Player();
        lower.setDiscount(0.9);
        lower.setLearning_rate(0.1);
        lower.setEpsilon(0.1);
        lower.setPayoff_matrix(4, 5, -3, -2);

        higher = new Player();
        higher.setDiscount(0.1);
```

```
        higher.setLearning_rate(0.1);
        higher.setEpsilon(0.1);
        higher.setPayoff_matrix(4, -3, 5, -2);

        initial_state = (int) (Math.random() * 4);
        current_state = initial_state;
        states_results[0][t][current_state]++;

        for (i = 1; i < episodes_before; i++) {

          lower.setCurrent_action(lower.choose_action(current_state));
          higher.setCurrent_action(higher.choose_action(current_state));


          next_state = get_state(lower.getCurrent_action(), higher.getCurrent_action());

          lower.update_Q(current_state, next_state);
          higher.update_Q(current_state, next_state);

          if ( flag==0 && i%payoff_interval==0 &&
               ((T+ratio_T) > (R+ratio_R+0.01)) &&
               ((R+ratio_R) > (P+0.01)) &&
               ((P) > (S+0.01)) &&
               (2*(R+ratio_R) > ((T+ratio_T) + (S)))
               ){
            R = R + ratio_R;
            T = T + ratio_T;

            higher.setPayoff_matrix(R, S, T, P);
            lower.setPayoff_matrix(R, T, S, P);

            System.out.printf("R:%.4f T:%.4f S:%.4f P:%.4f e:%d t:%d\n", higher.getPayoff_matrix(0),
                 higher.getPayoff_matrix(2), higher.getPayoff_matrix(1),
                 higher.getPayoff_matrix(3), i, t);
          }
          // Set Matrix back to standards value
          if (i % standards_interval == 0){
            R = 4;
            T = 5;
            S = -3;
            P = -2;
            lower.setPayoff_matrix(4, 5, -3, -2);
            higher.setPayoff_matrix(4, -3, 5, -2);
            System.out.printf("R:%.4f T:%.4f S:%.4f P:%.4f e:%d t:%d\n", higher.getPayoff_matrix(0),
                 higher.getPayoff_matrix(2), higher.getPayoff_matrix(1),
                 higher.getPayoff_matrix(3), i, t);
          }

          current_state = next_state;

          update_states_results(i, current_state, t);
          update_payoff_results(i, lower.getPayoff_matrix(current_state),
higher.getPayoff_matrix(current_state), t);
        }
```

```
        // Set epsilon = 0 and initial payoff matrix
        lower.setEpsilon(0);
        higher.setEpsilon(0);
        R=4; T=5; S=-3; P=-2;
        lower.setPayoff_matrix(4, 5, -3, -2);
        higher.setPayoff_matrix(4, -3, 5, -2);

        for (i = 1; i <= episodes_after; i++) {
          lower.setCurrent_action(lower.choose_action(current_state));
          higher.setCurrent_action(higher.choose_action(current_state));

          next_state = get_state(lower.getCurrent_action(), higher.getCurrent_action());

          lower.update_Q(current_state, next_state);
          higher.update_Q(current_state, next_state);

          if ( flag==0 && i%payoff_interval==0 &&
               ((T+ratio_T) > (R+ratio_R)) &&
               ((R+ratio_R) > (P)) &&
               ((P) > (S)) &&
               (2*(R+ratio_R) > ((T+ratio_T) + (S)))
               ){
            R = R + ratio_R;
            T = T + ratio_T;
            higher.setPayoff_matrix(R, S, T, P);
            lower.setPayoff_matrix(R, T, S, P);
            System.out.printf("R:%.4f T:%.4f S:%.4f P:%.4f e:%d t:%d\n", higher.getPayoff_matrix(0),
                 higher.getPayoff_matrix(2), higher.getPayoff_matrix(1),
                 higher.getPayoff_matrix(3), i, t);
          }
          // Set Matrix back to standards value
          if (i % standards_interval == 0){
            R = 4;
            T = 5;
            S = -3;
            P = -2;
            lower.setPayoff_matrix(4, 5, -3, -2);
            higher.setPayoff_matrix(4, -3, 5, -2);
            System.out.printf("R:%.4f T:%.4f S:%.4f P:%.4f e:%d t:%d\n", higher.getPayoff_matrix(0),
                 higher.getPayoff_matrix(2), higher.getPayoff_matrix(1),
                 higher.getPayoff_matrix(3), i, t);
          }

          current_state = next_state;

          update_states_results(episodes_before+i-1, current_state, t);
          update_payoff_results(episodes_before+i-1, lower.getPayoff_matrix(current_state),
higher.getPayoff_matrix(current_state), t);
        }
      }
      statistics(trials, (episodes_before+episodes_after));
      printToFile_statistics(trials, (episodes_before+episodes_after));
    }

}
```

Appendix B

Class Player.java

```
/**
 * Created by jeannettechahwan on 20/01/15.
 */

// Aristedou Player
public class Player {
   int counter=0;
   private double learning_rate;
   private double epsilon;
   private double discount;
   private double[][] Q_table;
   private double[] payoff_matrix;
   private int current_action;

   public Player(){
      this.Q_table = new double[4][2];
      this.payoff_matrix = new double[4];
   }

   public void setDiscount(double value){
      this.discount = value;
   }

   public double getDiscount(){
      return this.discount;
   }

   public void setEpsilon(double value){
      this.epsilon = value;
   }
   public double getEpsilon(){
      return this.epsilon;
   }

   public void setLearning_rate(double value){
      this.learning_rate = value;
   }

   public double getLearning_rate(){
      return this.learning_rate;
   }

   public void setQ_table(int row, int column, double value){
      this.Q_table[row][column] = value;
   }

   public double getQ_table(int row, int column){
      return this.Q_table[row][column];
```

```
    }

    public void setPayoff_matrix(double value1, double value2, double value3, double value4){
        payoff_matrix[0] = value1;
        payoff_matrix[1] = value2;
        payoff_matrix[2] = value3;
        payoff_matrix[3] = value4;
    }

    public double getPayoff_matrix(int row){
        return this.payoff_matrix[row];
    }
    public void setCurrent_action(int action){
        this.current_action = action;
    }

    public int getCurrent_action(){
        return this.current_action ;
    }

    public int choose_action(int state){
        int action;
        double random = Math.random();

        if(random<epsilon){          //explore
            action = (int) (Math.random() * 2);    // values: 0(cooperate) or 1(defect)
        }
        else {    //exploit
            // find best known action
            if (this.getQ_table(state, 0) > this.getQ_table(state, 1))
                action = 0;
            else if(this.getQ_table(state, 0) < this.getQ_table(state, 1))
                action = 1;
            else
                action = (int) (Math.random() * 2);    // values: 0(cooperate) or 1(defect)
        }
        return  action;
    }

    public void update_Q(int current_state, int next_state){
        double max, Q;

        // find maximum value from Q table
        if(this.getQ_table(next_state,0) >= this.getQ_table(next_state,1))
            max = this.getQ_table(next_state, 0);
        else
            max = this.getQ_table(next_state, 1);

        // Calculate Q and set value in player's Q table
        Q = ((1 - this.learning_rate) * this.getQ_table(current_state, this.getCurrent_action()))
              + (this.learning_rate * (this.getPayoff_matrix(next_state) + (this.getDiscount() * max)));

        this.setQ_table(current_state, this.getCurrent_action(), Q);
    }
}
```