

**CE4046 Intelligent Agents: Assignment 2**

**Repeated Three Prisoner’s Dilemma**

James Tan Chee Min

U1720400A

School of Computer Engineering

Nanyang Technological University

14th April, 2019

**Abstract**

*To design an agent that performs well in a three-player prisoner’s dilemma*

Contents

[Introduction 2](#_Toc6258885)

[Payoff Matrix 2](#_Toc6258886)

[Payoff Hierarchy 2](#_Toc6258887)

[Social Welfare Hierarchy 2](#_Toc6258888)

[Implementation 3](#_Toc6258889)

[Pseudocode 3](#_Toc6258890)

[Code 3](#_Toc6258891)

[Decision Breakdown 4](#_Toc6258892)

[Evaluation 5](#_Toc6258893)

[Player Case Study 5](#_Toc6258894)

[Nice Player 5](#_Toc6258895)

[Nasty Player 5](#_Toc6258896)

[Random Player 5](#_Toc6258897)

[Tolerant Player 5](#_Toc6258898)

[Freaky Player 5](#_Toc6258899)

[Tit-For-Tat Player 5](#_Toc6258900)

[Benchmarks 6](#_Toc6258901)

[Example Players Pool 6](#_Toc6258902)

[GitHub and Popular Players Pool 6](#_Toc6258903)

[Mass Pool 6](#_Toc6258904)

[Conclusion 6](#_Toc6258905)

# Introduction

In a game of prisoner’s dilemma, the dominant strategy for players would be to defect. However, as we change it into a repeated prisoner’s dilemma whereby the prisoners’ have memory of previous prisoners’ choices, this phenomenon may very well be removed and instead will encourage cooperation among agents.

We are tasked to design an agent that performs well in a three-player prisoner’s dilemma. This adds complexity to the situation as now the agent has to consider both agents and strike a balance between both strategies.

## Payoff Matrix

The payoff matrix is designed in a way that, similarly to a two-player prisoner’s dilemma, optimal social welfare is achieved when all agents cooperate. Below shows the payoff matrix of agent 1 given the actions of all the agents.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Agent 2 Cooperates | | Agent 2 Defects | |
| Agent 3 Cooperates | Agent 3 Defects | Agent 3 Cooperates | Agent 3 Defects |
| Agent 1 Cooperates | 6 | 3 | 3 | 0 |
| Agent 1 Defects | 8 | 5 | 5 | 2 |

*Payoff Matrix in a Three Player Prisoner’s Dilemma*

## Payoff Hierarchy

Below shows the hierarchy of the payoff based on actions of the agents. C represents cooperation by the agent, D represents defect by the agent. *U(DDC)* represents defection by agent 1 and only either one of agent 2 or agent 3 chose to defect.

*Hierarchy of Payoff based on Agent’s Actions*

## Social Welfare Hierarchy

Below shows the hierarchy of social welfare based on the actions of agents. C represents cooperation by the agent, D represents defect by the agent. U(DDC) represents defection by 2 agents and cooperation by 1 agent.

*Hierarchy of Social Welfare based on Agent’s Actions*

# Implementation

According to the payoff matrix and hierarchy, the maximum private utility can be achieved by being the sole defector, while the maximum social welfare can be achieved by having everyone cooperating. Tit-For-Tat is given as an example and is a good choice that maximises social welfare as it maximises social welfare and punishes those who defects and will be the basis of the agent.

## Pseudocode

*IF first round*

*CHOOSE cooperate*

*ELSE IF first 4 rounds AND someone has defected in the previous round*

*CHOOSE defect*

*ELSE IF every 10 rounds AND one of the opposing agents is nasty*

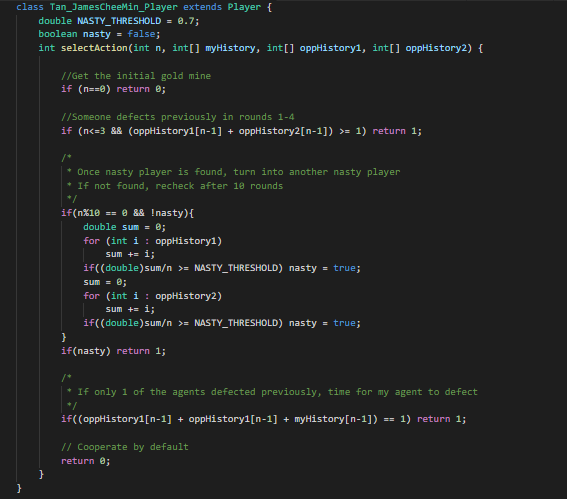
*CHOOSE defect infinitely*

*ELSE IF previously only one agent defected  
 CHOOSE defect*

*ELSE*

*CHOOSE cooperate*

## Code



## Decision Breakdown

|  |  |
| --- | --- |
| Decision 1 | if (n==0) return 0; |
| Breakdown | Cooperate initially, does not trigger Tit-For-Tat to defect. |

|  |  |
| --- | --- |
| Decision 2 | if (n<=3 && (oppHistory1[n-1] + oppHistory2[n-1]) >= 1) return 1; |
| Breakdown | Tit-For-Tat for the first 4 rounds, manipulates other agents into thinking this is a Tit-For-Tat agent, counters probing agents. |

|  |  |
| --- | --- |
| Decision 3 | if (n%10 == 0 && !nasty){  double sum = 0;  for (int i : oppHistory1)  sum += i;  if((double)sum/n >= NASTY\_THRESHOLD) nasty = true;  sum = 0;  for (int i : oppHistory2)  sum += i;  if((double)sum/n >= NASTY\_THRESHOLD) nasty = true;  }  if (nasty) return 1; |
| Breakdown | Every 10 rounds, check if one of the opposing agents is a nasty player that chooses to defect majority of the time, counters nasty agents and probing agents. |

|  |  |
| --- | --- |
| Decision 4 | if((oppHistory1[n-1] + oppHistory1[n-1] + myHistory[n-1])==1) return 1; |
| Breakdown | If only one of the agents defected previously, choose to defect. Exploits random-target based three-player Tit-For-Tat. |

|  |  |
| --- | --- |
| Decision 5 | return 0; |
| Breakdown | Cooperate by default to promote other Tit-For-Tat agents to cooperate as well. |

# Evaluation

James’ agent ultimate aim is to cooperate indefinitely to achieve *U(CCC),* if any of the agents defect, the agent will try to encourage other agents to cooperate to achieve *U(CCC),* but if given the opportunity, the agent will defect to achieve *U(DCC).* If a nasty player is onboard, *U(DDC)* or *U(DDD*) will be achieved. With this strategy, the agent will try to achieve maximum social welfare, if not possible, be the highest scoring amongst the 3 agents.

## Player Case Study

As this is a three-player prisoner’s dilemma, it is assumed that the 3rd agent will either be the target study agent or James’ agent.

### Nice Player

Nice Player cooperates indefinitely

Result: As Nice Player cooperates indefinitely; James’ agent will also cooperate as it does not meet the requirements to fulfil decision 1-4. Achieving *U(CCC).*

### Nasty Player

Nasty Player defects indefinitely

Result: As Nasty Player defects indefinitely; James’ agent will cooperate for the first round based on decision 1, defect for the next 3 rounds based on decision 2, defect/cooperate based on agent pool based on decision 4 and 5, defect from round 11 onwards based on decision 3. Achieving *U(DDD)* or *U(DDC)* for majority of the time.

### Random Player

Random Player randomly cooperates or defects.

Result: As Random Player randomly cooperates or defects, due to its randomness, it is difficult to estimate the outcome. Every outcome is possible, with *U(CDC)* being the slight majority as the agent tries to promote cooperation in decision 5.

### Tolerant Player

Tolerant Player cooperates unless more than half of opponents’ actions are to defect.

Result: As James’ agent does not defect, both will cooperate indefinitely as it does not meet the requirements to fulfil decision 1-4. Achieving *U(CCC).*

### Freaky Player

Freaky Player chooses to be a nice player or nasty player at the start.

Result: Achieves *U(CCC)* if freaky chooses to be nice, *U(DDD)* or U*(DDC)* if freaky chooses to be nasty. Refer to results above.

### Tit-For-Tat Player

Tit-For-Tat chooses a random opponent’s previous action as its action

Result: James’ agent will cooperate as it does not meet the requirements to fulfil decision 1-4, Tit-For-Tat will do the same too. Achieves *U(CCC)*.

## Benchmarks

As there are some randomness in the player’s actions, a sample size of 10,000 is used to determine the average expected performance of James’ agent. Multiple simulations are done with different pool of agents. Below shows the different pool of agents used for simulation.

### Example Players Pool

The example players above and James’ agent.

### GitHub and Popular Players Pool

Agents from GitHub[[1]](#footnote-1) are sourced, agents are developed[[2]](#footnote-2) for three-player prisoner’s dilemma from popular pre-existing strategies[[3]](#footnote-3) HardProber, Tideman and Chieruzzi, Generous Tit-For-Tat and NaiveProber and James’ agent.

### Mass Pool

All agents included above.

*Performance of James’ Agent in Various Pools*

# Conclusion

Overall, agents should cooperate as much as possible to increase overall social welfare. Many good performing strategies utilise Tit-For-Tat as a building block as it awards cooperation and punishes defection. The implemented strategy is a modified version of Tit-For-Tat that performs well in most situations.

1. almighyGOSU, <https://github.com/almightyGOSU/CZ4046-Intelligent-Agents-Assignment-2> (April, 2016) [↑](#footnote-ref-1)
2. James Tan, <https://github.com/lolfuljames/CE4046-ThreePrisonersDilemma> (April, 2019) [↑](#footnote-ref-2)
3. Axelrod, <https://axelrod.readthedocs.io/en/stable/reference/overview_of_strategies.html> (2015) [↑](#footnote-ref-3)