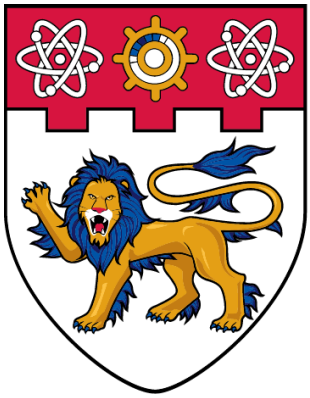

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC4003 Intelligent Agents

Assignment 2

Derrick Ng Choon Seng

U2122873F

Table of Contents

1. Introduction

A 3 player repeated Prisoners' Dilemma is more complex than a 2 player version. There are 3 possible scenarios in a 3 player Prisoners' Dilemma game:

1. Both opponents cooperate
2. Both opponents defect
3. One opponent cooperates while the other defects

Actions	Payoffs
C, C, C	6, 6, 6
C, C, D	3, 3, 8
C, D, C	3, 8, 3
C, D, D	0, 5, 5
D, C, C	8, 3, 3
D, C, D	5, 0, 5
D, D, C	5, 5, 0
D, D, D	2, 2, 2

Table 1: Payoffs for 3 player Prisoners' Dilemma

From Table 1, we can see that the best action is to defect when the other 2 players cooperate. However, it is unlikely that the opponents will keep cooperating while we defect. As such, the next best action would be for everyone to cooperate. It is also important to not be exploited by other players who keep defecting, as cooperating while others defect would lead to the worst outcome.

2. Pilot study and Assumptions

Initial player models included a Bayesian opponent modelling player which aims to predict the type of player the opponents are and decides the best action to take based on the opponents' expected actions. Another model included a Q learning player.

However, after evaluating the players and considering the game environment, it was decided that both agent players would not be used. Both players are inconsistent and tend to place around 3rd or 4th place. This could be due to the training required by the agent before being able to perform well, and since the payoffs are accumulated, the performance during the training phase would affect the overall performance. The Bayesian opponent modelling agent also requires prior knowledge of the types of opponents there are, but we do not know what kind of players other students would design.

Instead, a simple rule-based agent was designed. An assumption was made that due to the limited actions and outcomes, there are not many different strategies and that players designed by other students would be variants of the existing players provided, mainly the T4T player.

3. Agent Design

The player should always seek cooperation, however when faced with opponents who try to exploit our cooperation, the player would retaliate, as per the T4T strategy. We make use of a variant of the T4T strategy. We try to be forgiving to seek and encourage cooperation, however we should be reactive and retaliate against consistent defects from opponents. Instead of retaliating immediately, the player would look at the opponents' past moves withing a certain window and decide whether to retaliate. Finally, when the game is reaching the end, the player would exploit other players by defecting unexpectedly to try and gain a slight advantage and increase in payoffs.

3.1. Rules to follow

3.1.1. Rule 1

The first rule that the agent follows is that it should always cooperate regardless of the opponents' actions in the first *k* rounds. This is to first create an environment for cooperation with the opponents and encourage them to cooperate as well.

```
// cooperate in first k rounds
if (n < k) {
    return 0;
}
```

Figure 1: Code snippet corresponding to rule 1

3.1.2. Rule 2

The second rule that the agent follows is that it should always defect after *m* rounds, when the game is about to end. Since the game is about to end, there is no need for the player to continue to cooperate and should try to exploit the opponents last minute for a slight advantage.

```
// Defect the last m rounds onwards
else if (n >= m) {
    return 1;
}
```

Figure 2: Code snippet corresponding to rule 2

3.1.3. Rule 3

The third rule is that the agent should seek to exploit others when possible. In scenario 1 (mentioned in Section 1), where both opponents cooperated, the agent would look at several rounds prior to the cooperation and decide whether to exploit them by defecting.

Given that the opponents cooperated in round n-1, the agent would look at *c* rounds before cooperation (round n-2 and before). In this case, there are a few possibilities

A. Both opponents cooperated all rounds
B. Either opponent cooperated at least once within all rounds
C. At least one opponent defected all rounds

In scenarios B and C, the agent would continue to seek and encourage cooperation since there is still an opponent that defects and exploitation might only lead to 5 points instead of 8 points. However, in scenario A, where both opponents display consistent cooperation, the agent would seek to exploit them by defecting for 1 round to gain a slight advantage in payoffs.

```
// Both opponents cooperate
if (oppHistory1[n-1] == 0 && oppHistory2[n-1] == 0) {
    int action = 0;
    int persistentCoop1 = 0;
    int persistentCoop2 = 0;

    for (int i = 1; i <= c; i++) {
        if (oppHistory1[n-1-i] == 0) { // If opponent 1 cooperated all last c rounds before cooperating
            persistentCoop1++;
        }
        else if (oppHistory2[n-1-i] == 0) { // If opponent 2 cooperated all last c rounds before cooperating
            persistentCoop2++;
        }
    }
    // If both opponents cooperated consistently
    if (persistentCoop1 >= c && persistentCoop2 >= c) {
        action = 1; // Defect to exploit them
    }
    else {
        action = 0; // Continue cooperating
    }

    return action;
}
```

Figure 3: Code snippet corresponding to rule 3

3.1.4. Rule 4

The fourth rule is that the agent should be forgiving but not be exploited. It should retaliate, but not immediately. In scenario 2 (mentioned in Section 1), where both opponents defected, the agent would look at several rounds prior and decide whether to retaliate or not, similar to rule 3 in the previous scenario.

Given that the opponents defected in round n-1, the agent would look at *d* rounds before defection (round n-2 and before). In this case, there are the same possibilities (A, B and C) as in scenario 1.

In scenarios A and B, the agent would forgive the opponents and continue to seek cooperation. In scenario C, the agent would punish the opponents by defecting for *p+1* rounds regardless of their subsequent actions. Since at least 1 opponent have been defecting consistently, we should defect for a few rounds to prevent exploitation by them. After the punishment is over, the agent would then return to normal and decide whether to cooperate or defect based on the rules.

```
// Both opponents defect
else if (oppHistory1[n-1] == 1 && oppHistory2[n-1] == 1) {
    int action = 0;
    int persistentDefect1 = 0;
    int persistentDefect2 = 0;

    for (int i = 1; i <= d; i++) {
        if (oppHistory1[n-1-i] == 1) { // If opponent 1 defected all last d rounds before defecting
            persistentDefect1++;
        }
        else if (oppHistory2[n-1-i] == 1) { // If opponent 2 defected all last d rounds before defecting
            persistentDefect2++;
        }
    }
    // If at least one opponent consistently defected
    if (persistentDefect1 >= d || persistentDefect2 >= d) {
        punishCounter = p; // p more rounds of punishment after current round
        action = 1;
    }
    else {
        action = 0;
    }

    return action;
}
```

Figure 4: Code snippet corresponding to rule 4

```
// If punishment is active, defect
if (punishCounter > 0) {
    punishCounter--;
    return 1;
}
```

Figure 5: Code snippet of punishment used in rule 4

3.1.5. Rule 5

The fifth rule is that the agent should seek cooperation outside of the rules stated above. For example, in scenario 3 (mentioned in Section 1), when there is at least one opponent cooperating, the agent should cooperate since there is another opponent willing to cooperate.

```
// Only one opponent defects
else {
    return 0;
}
```

Figure 6: Code snippet corresponding to rule 5

3.2. Fine tuning

Since the number of rounds ranges from 90-110 for each game, we are unable to determine a fixed number of rounds to start defecting from according to rule 2. Fine tuning was performed to determine hyperparameter values used in the various rules and are shown in Figure 7.

```
int k = 8;
int m = 105;
int c = 4;
int d = 1;
int p = 5;
```

Figure 7: Hyperparameter values after fine tuning

3.3. Overview

Based on the hyperparameters in Figure 7, the agent would cooperate no matter what in the first 8 rounds, then defect all the way from round 105 onwards if there are more than 105 rounds. Between those rounds, the agent would cooperate by default unless both opponents cooperated or both opponents defected (in round n-1). Upon cooperation, the agent would look at 4 rounds prior to the cooperation (rounds n-2 to n-5) and if both opponents have been consistently cooperating, exploit them by defecting to get a high payoff. Upon defection, the agent would then look at 1 round prior to the defection (round n-2) and if either opponent has been consistently defecting, a punishment of 6 consecutive defects would be metered out.

4. Evaluation

To evaluate the agent, the tournament/match against the provided players were ran 10 times consecutively, and the score and ranking of each player were recorded (rounded off to 2 decimal places). The average score of each player was then calculated and the

overall ranking of the agent was obtained. Table 3 shows the individual scores of each player and their overall performance in 10 runs (refer to Appendix for the code results).

		Players							
		me	nice	nasty	random	tolerant	freaky	t4t	ranking
Run number	1	167.77	158.66	153.13	165.33	163.2	161.62	168.1	2
	2	161.53	160.93	155.88	145.99	163.04	170.19	163.65	4
	3	167.46	148.59	155.79	157.99	165.96	172.54	169.1	3
	4	170.82	164.79	158.59	169.43	166.82	152.47	159.13	1
	5	166.84	169.75	159.11	154.83	166.72	150.69	162.28	2
	6	169.76	160.86	160.14	164.2	162.22	154.74	166.99	1
	7	166.3	164.37	150.25	156.07	168.76	162.16	171.52	3
	8	159.86	156.88	162.73	146.67	162.67	161.26	156.16	4
	9	166.12	163.6	163.51	150.69	158.03	160.21	166.93	2
	10	165	162.45	158.65	149.08	167.96	153.9	159.54	2
	avg/overall	166.146	161.088	157.778	156.028	164.538	159.978	164.34	1

Table 2: Individual and average scores of players

The agent places within the top 2 most of the time, while coming in 3rd or 4th occasionally. The agent has the highest average score as compared to the other players, showing its consistent performance over all the match runs. It is important to note however that its performance against the given players do not represent its actual performance in the test as we are competing against unknown players designed by other students.

5. Conclusion

In a 3 player Prisoners' Dilemma game, the best action would be for all players to cooperate. However, due to the limited rounds and information on players, complex strategies such as opponent modelling and reinforcement learning would unlikely perform well. Since we also do not know what other players would do, one can balance cooperation and exploitation while being reactive to prevent others from exploiting oneself.

6. Appendix

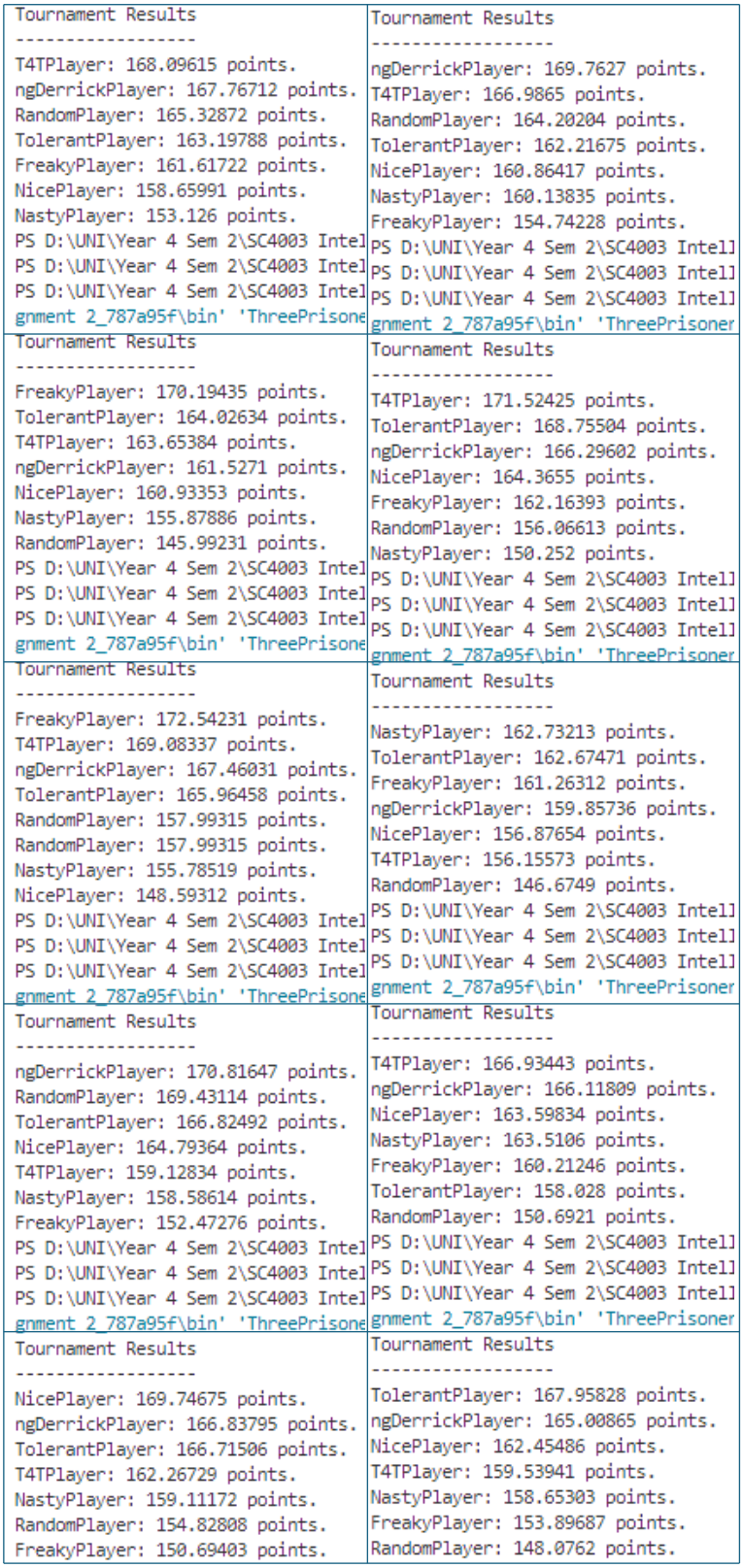

Figure 8: Results of 10 consecutive match runs