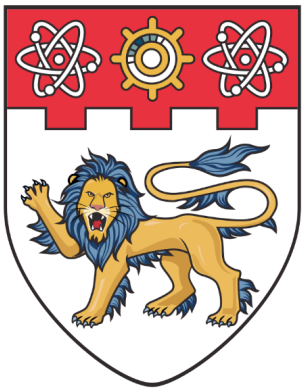

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4046 Intelligent Agents

Assignment 2

Name: Lim Kai Sheng

Matriculation No.: U2020321C

Table of Content

1. Introduction

a. Reward Structure

The following table lists all possible combinations along with their respective payoff. Cooperation is represented as “0”. While defection is represented as “1”.

Table 1: Payoff of Three-Player Prisoner’s Dilemma

You	Opponent 1	Opponent 2	Payoff
0	0	0	**6**
0	0	1	**3**
0	1	0	**3**
0	1	1	**0**
1	0	0	**8**
1	0	1	**5**
1	1	0	**5**
1	1	1	**2**

2. Agent Design

a. Issues

Why can’t we use Always-Defect strategy in Three-player Prisoner’s Dilemma

In a two-player Prisoner's Dilemma, an always-defect agent ensures a win or a tie. This is because the opponent would never be able to gain points from the current player.

However, in a three-player Prisoner's Dilemma, the dynamics can shift against an always-defect agent. This occurs when the other two players band together to punish the always-defecting agent by defecting. In other words, the always-defect agent has a chance to win, but we (the other opponents) will make sure you don't get a large reward.

As a result, it is still critical to collaborate with other players whenever possible.
So that the agent can earn the highest possible reward while attempting to win each round

Defection should remain the preferred option.

Table 2: Payoff in descending order

You	Opponent 1	Opponent 2	Payoff
1	0	0	**8**
0	0	0	**6**
1	0	1	**5**
1	1	0	**5**
0	0	1	**3**
0	1	0	**3**
1	1	1	**2**
0	1	1	**0**

According to the table, the best action is still defection, which is highlighted by the green rows. However, there is a fine line between encouraging cooperation to gain more points and always-defect to win the game.

b. Trialed strategies

i. Simple Majority (Rank 1st)

Start by cooperating. Continue to cooperate as long as the opponents cooperate more than or equal to the number of times they have defected. Otherwise, the agent defects.

ii. Tit-for-Two-Tats (Rank 2nd)

Start by Cooperating. Defect only when the opponent defects two times.

iii. Gradual (Rank 3rd)

Start by cooperating. Continue to cooperate as long as the opponent cooperates. After the first defection of the other player, it defects once and cooperates twice; In other words, it reacts to the opponent's defection and then calms down its opponent with two cooperations.

c. Thought Process

i. Rule 1: Start by cooperating

The idea is to start by being friendly. This is to prevent triggering punishing strategies as the start. Remember, the early stages of the rounds are to cooperate as much as possible to maximize the payoff returns. It is okay to take a loss of utility in the first round than to trigger soft-grudger or grim trigger strategies.

ii. Rule 2: Defect for the last remaining rounds

This helps to optimize our agent's total payment. It is based on a game-theoretic analysis that reveals that "always defecting" is the dominating strategy. As the game nears its conclusion, the time remaining in mutual business with the opponents shortens. As a result, for maximum gain, our agent is designed to exploit the system by constantly defecting regardless of the opponent's actions as the match nears its conclusion.

iii. Rule 3: Incorporation of Tit-for-twice-tat

The plan is to cooperate on the first move and defect only if both opponents defect twice. This restriction exists because there may be methods that encourage the opponent to move to always-defect mode too soon. For example, tit-for-tat in conjunction with always-defect, or strategy like the grim trigger. I should not be hesitant to defect and penalize the other players in such a case. Otherwise, I'll be on the losing end because rule 4 takes a few rounds to activate.

iv. Rule 4: Incorporation of Soft Majority

The plan is to cooperate on the first move and continue to cooperate as long as the number of times the opponent has cooperated is greater than the number of times it has defected, or else it defects. This is akin to the “tolerant player” found in the template source code. However, based on experimentation, I have tuned the threshold to a higher value (0.6). This strategy is helpful because our agent recognizes the importance of mutual cooperation in maximizing its long-term benefits without being taken advantage of.

d. Implementation

Table 3 : Code snippet of implemented strategy

```
class LIM_KAISHENG_Player extends Player {
   // Helper function for TitForTwoTats
   boolean defected_twice(int[] history) {
       int count = 0;
       int length = history.length;

       if (length < 2)
           return false;

       for (int i = length - 2; i < length; i++) {
           if (history[i] == 1) {
               count++;
           }
       }

       if (count == 2)
           return true;
       else
           return false;
   }

   // Helper function for SoftMajority
   float cooperate(int[] history) {
       int count = 0;
       int length = history.length;

       for (int i = 0; i < length; i++)
           if (history[i] == 0)
               count++;

       return (float) count / length;
   }

   int selectAction(int n, int[] myHistory, int[] oppHistory1, int[] oppHistory2) {
       /* 1. Cooperate in first round */
       if (n == 0)
           return 0;

       /* 2. Closing to the end of the game, defect to maximize rewards.
       Set N to a smaller value assuming most other players will do this trick as well */
       if (n > 90) {
           return 1;
       }

       /* 3. Tit-For-Twice-Tats
       Cross over to defection once both players had defected twice defecting */
       boolean opp1 = defected_twice(oppHistory1);
       boolean opp2 = defected_twice(oppHistory2);

       if (opp1 && opp2) {
           return 1;
       }
```

```
        /* 4. SoftMajority Strategy
        With percentage tuned higher */
        float percentage1 = cooperate(oppHistory1);
        float percentage2 = cooperate(oppHistory2);
        if (percentage1 >= 0.6 && percentage2 >= 0.6){
            return 0; // Cooperate since opponents are cooperative
        }
        else{
            return 1; // Defect since opponents are defecting
        }
    }
```

3. Evaluation

The runTournament() was modified to reflect a more accurate / true result of the strategy. Within each iteration, the agents are still competing 100 times, but it is further encapsulated to run another 1000 times.

Table 4: The overall scoring and ranking of the implemented source code

Implemented Strategy v.s. Default Src Code	Implemented Strategy v.s. Senior's Agents
1. LIM_KAISHENG_Player : 26236.902 points. 2. SoftMajority : 26161.188 points. 3. TitForTwoTats : 25985.477 points. 4. T4TPlayer : 25626.871 points. 5. TolerantPlayer : 25588.26 points. 6. NicePlayer : 23617.215 points. 7. FreakyPlayer : 21209.355 points. 8. NastyPlayer : 19131.832 points. 9. RandomPlayer : 18962.402 points. N.b. To illustrate the improvement of strategy in doing a hybrid of SoftMajority and TitForTwoTats.	1. LIM_KAISHENG_Player : 31997.455 points. 2. Javelin_Player : 31834.941 points. 3. Junyuan_Player : 31828.443 points. 4. SoftMajority : 31761.188 points. 5. TitForTwoTats : 31737.838 points. 6. TolerantPlayer : 31259.836 points. 7. T4TPlayer : 31229.893 points. 8. NicePlayer : 28647.137 points. 9. FreakyPlayer : 25048.383 points. 10. NastyPlayer : 21881.816 points. 11. RandomPlayer : 20646.758 points. N.b. Seniors' codes are retrieved from public github repository. To illustrate performance in prototyping.

It is also worth noting that the overall ranking does change slightly because there's Freaky and Random Player. Both of which do randomize and result in slight deviation. However, the finding here is repeated on multiple iterations and is a good representation of the overall strategies' performances.

4. Conclusion

To summarize, designing an always-winning strategy is a challenging endeavor owing to the numerous ways an opponent may devise to counteract one's strategy.

In my approach to this assignment, I did a search to find our past projects that made a comparison of the various strategies and extracted 3 of which that I deemed promising. Following this, I did a more in-depth analysis and fine-tuned the parameters to further optimize the final strategy.

Here, our agent starts by cooperating to prevent triggering "grim trigger" or any Tit-for-tat.

Our agent will then strategically reciprocate cooperation and defection. It uses the Simple majority strategy to gauge the opponents' sentiments and cooperate if they are sincere. At the same time, staying engaged to any sign of defection with the Tit-for-twice-tat strategy.