International Journal of Multimedia and Ubiquitous Engineering
Vol.11, No.2 (2016), pp.309-322
http://dx.doi.org/10.14257/ijmue.2016.11.2.30

Agent Based Performance Analysis of Strategic Algorithms in Prisoner's Dilemma

Aastha Yadav, Chandini Bhambhani, Pronay Peddiraju
Ronnie D. Caytiles* and N.Ch. S.N. Iyengar

SCSE, Vellore Institute of Technology University, Vellore-632014, TN, India
**Department of Multimedia Engineering, Hannam University, Korea,*
{aasthay1705, chandini.bhambhani, pronay.y2k, rdcaytiles}@gmail.com,
nchsniyr@vit.ac.in

Abstract

To create a system that provides a comparison of multiple algorithms that may be tested in the Prisoner's Dilemma decision problem using two subjects in a dual agent environment. As an addition to understanding the effects of various algorithms and logic that helps influence a single agent's decision, our system aims at analysing the performance of the same algorithms in iterative and multi agent systems. The results are obtained by using concepts of Swarm Intelligence, Multiple Agent Systems and Super Agents within the testing system. The results of the research are to expose the advantages and disadvantages of each schema to help plan investments, predict outcomes and for real world application of the Prisoner's Dilemma in fields of Environmental Sciences, Psychology, Economics and many more such fields.

***Keywords**: Swarm Intelligence, Super-Agent, Multiple Agents*

1. Introduction

The well-known prisoner's dilemma game has become the classic economic example to demonstrate non-cooperative behavior: Two contestants face a ''dilemma'' in which, independent of each other's action, each player is better off by defecting than by cooperating. However, the outcome obtained when both defect, is worse for each player than the outcome both would have obtained if they would cooperate. Thus, self-interest oriented behavior does not lead to a globally optimal solution in all cases. A common view is that this puzzle illustrates a connection between individual and group rationality. Two players who both pursue rational self-interest may end up worse off than if both act contrary to rational self-interest.

This paper analyzes choice of outcome done among one of the four possible permutations in the canonical Prisoner's Dilemma payoff matrix.

Analysis of the strategy depends on the following factors of the strategy:

1. "Nice" nature of strategy
2. Retaliation factors
3. Forgiving Nature
4. Non-envious quality

Players can communicate with each other and hence have the possibility to play with each other, and thereby get to know each other in two stages of pre-play. In order to make the best choice, each player would have to know what the other player might do, but the structure of prisoner's dilemma prohibits players from having such knowledge, unless the situation or game is iterated. The prisoner's dilemma problems lack a single optimal strategy and both parties rely on each other to achieve most favorable results. When understood properly, this dilemma can multiply into hundreds of other more complex dilemmas.

The mechanisms that drive the prisoner's dilemma are the same as those faced by marketers, military strategists, poker players, and many other types of competitors. The simple models used in the prisoner's dilemma afford insights on how competitors will react to different styles of play, and these reactions will reveal suggestions on how those competitors will probably act in the future. Plethora of disciplines have studied the game, including artificial intelligence, biology, business, mathematics, philosophy, sociology, and political science.

2. Prisoner's Dilemma Problem

The Prisoner's dilemma relies on the existence of two extremes in terms of conditions. The entity or agent in our case may choose to betray or cooperate. A model based on rationality where entities forecast how the game would be played if they formed a coalition and then their ability to maximize their forecast, has been shown to make better predictions of the rate of cooperation in this and similar scenarios. The prisoner's dilemma, of course, is one of the most studied games in the literature. The rich background of prisoner's dilemma research allows us to contrast our results, about behavior of human subjects playing quantum games, with behavior in the classical version of the game, which is widely known. Furthermore, the prisoner's dilemma is also a simple version of the public goods game, for which a quantum mechanics performs efficiently for large groups. Specifically, for two players with equal preferences and endowments, the public goods problem reduces to the prisoner's dilemma. The prisoner's dilemma illustrates the free-rider problem in the simplest context of a two-person game, in which each player has the choice to "cooperate" or "defect". Payoffs for both players are higher when both of them choose to cooperate instead of both defecting. However, each individual is better off by defecting. The prisoner's dilemma involves the possibility of altruistic behaviors in which participants can either select actions that most benefit themselves or those that benefit the group as a whole but at some individual loss.

3. Literature Review

Prisoner's dilemma is still a current research area with nearly 15000 papers published over the last two years [Source: Google Scholar]. New strategies are developed and old ones revised for implementation in new areas. Research reveals plenty of existing classical economic mechanisms that solve the free-rider problem in different environments [5]. Another conclusion drawn was that cooperation is not the outcome in the infinitely repeated Prisoner's Dilemma [13].

New approaches upgrade known ideas through genetic algorithms and heuristic approaches and successfully recognize opponents, to anticipate their moves and try to achieve better results. These approaches have analyzed cooperative behavior in a prisoner's dilemma game in the presence of high stakes, communication, and two rounds of pre-play, involving two voting decisions. It is observed that stake size, communication as well as pre-play have a significant impact on cooperation [4].

Cooperative play in prisoner's dilemma games by designing an experiment to evaluate the ability of two leading theories of observed cooperation namely, reputation building and altruism have been studied. They analyze both one-time and finitely repeated games to gauge the importance of these theories. We can conclude that neither altruism nor reputation building alone can explain our observations [9]. Complex adaptive systems to find the optimal approach do not aggregate strategies in hope of demonstrating the "Wisdom of crowd" phenomenon [6].

There is always a problem of possibility to misjudge the opponent, which will bring worse results in the end. However, the information carries the key role in any sort of intelligent activities and strategies. Individuals with more information will have advantage in most situations as the strategies that learn about the opponents and adjust their own responses will certainly have an increasingly important role in the future. Hence, our project aims at bringing in a new viewpoint using software instead of hardware in optimizing the strategies or moves in a prisoner's dilemma, with the help of performance analysis of various algorithms on a set of agents.

4. Derivation of Optimal Strategy

1. Bayesian Nash Equilibrium: If the statistical distribution of opposing strategies can be determined, an optimal counter strategy can be derived analytically.
2. Monte Carlo Simulation: Simulation of large population of entities where entities with low scores die out and those with high scores reproduce (induce the instantiation of more such entities). Mix of algorithms in final population generally depend on the mix in the initial population.

5. Architecture

There are a lot of different architectures we can choose when playing social dilemma games, or in an actual social dilemma with iterative processes. In our project, our aim is to achieve a general case involvement of single agent-to-agent interaction or multi-agent interaction, which runs through a set of algorithms to produce an optimized result, which leads to the benefit of either single or both parties as required by the user.

Table 1. Representation of Prisoners' Dilemma

		Team B	
		Cooperate	**Defect**
Team A	**Cooperate**	R,R	S,T
	Defect	T,S	P,P

5.1 General Case

The general case in our project is for multi-agents. Single agent-to-agent interaction has been explained in detail as a special case. In multi-agent interaction, there are single agent-to-agent interactions taking place simultaneously, as in, at a time there can be 'n' number of interactions between only 2 agents. The architecture of the general case is explained by the following Figure 1.

If both players cooperate, they both receive the reward R for cooperating. If both players defect, they both receive the punishment payoff P. If say agent A defects while B cooperates, then A will receive the temptation payoff T, while B receives the non-beneficial payoff S. If agent B defects while A cooperates, then B will receive the temptation payoff S, while B receives the non-beneficial payoff T.

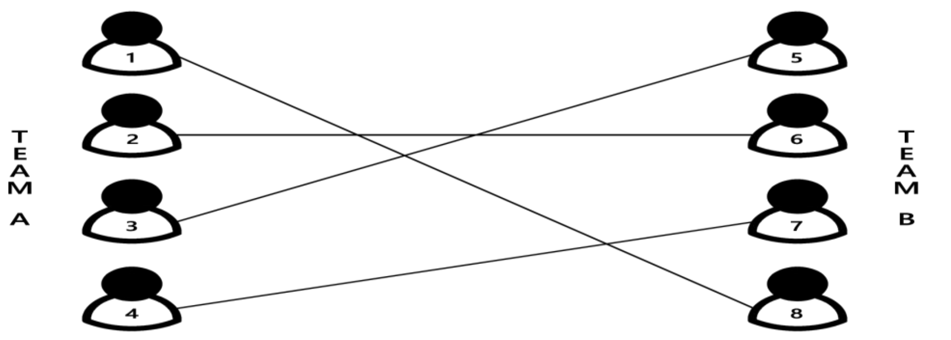


Figure 1. General Architecture

Here n = 4, *i.e.*, the number of interactions between two agents in the above multi-agent interaction is 4. Our implementation of code will include various algorithms wherein, the agents are designed to carry out one of the strategies as mentioned in the list of strategies explained below. Once the above interactions have taken place, only those interacting agents in a relationship, who have benefitted will stay on. Say for example, Agent 1 benefits over Agent 8 ensures Agent 1 stays on, Agent 2 and Agent 6 benefit via cooperation ensuring both Agent 2 and Agent 6 stay on, Agent 3 and Agent 5 get non-beneficial results which eliminated both Agent 3 and Agent 5, and Agent 7 benefits over Agent 4 which results in eliminating Agent 4.The non-benefitted agents are eliminated to serve their time or deal with their punishment but, the other agents are paired with remaining agents on either side until one or both teams emerge with beneficial payoffs.

5.1.1 Case 1

If the user requires only two participating agents, a single agent-to-agent interaction will entail where n=1. This is a particular case of the general case stated above, which has a faster compile time and can be specified by the user to eradicate multiple agent interaction complexities. Once again, strategies are implemented with both interacting Agents and one or both agents receive beneficial payoffs.

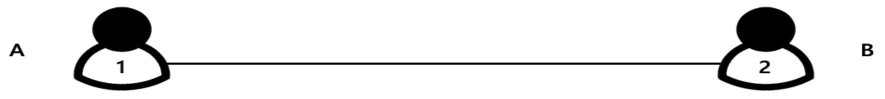


Figure 2. n=2 Agents

5.1.2 Case 2

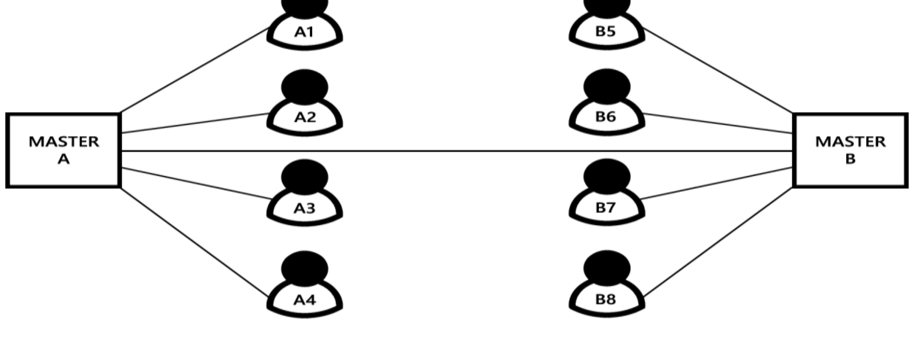


Figure 3. Master Slave Implementation

This has the same as the general case, consisting of ‘n’ interactions, which is reduced to an optimal solution, but here, we have a master agent. The master agent in each team is the agent that all other subordinate agents try to protect. When involved in interactions with opposing agents, they sacrifice themselves by settling for non-beneficial payoffs so that the master agent always receives the reward payoff. By following this mechanism, the master agent earns enough reward payoffs to benefit the entire team.

6. Working Implementation

In this project, primarily agents will be made using the agent capabilities of the C# programming language using Visual C++ programming and the Visual Studio software. The agents were made using the available agent classes under the header file #include <agents.h> and concurrency class in the VC++ system32 console application files. Once the agents were created, we applied the existing strategic algorithms on them to arrive at the most optimized and efficient algorithm as seen in the following two models:

Flow Diagram

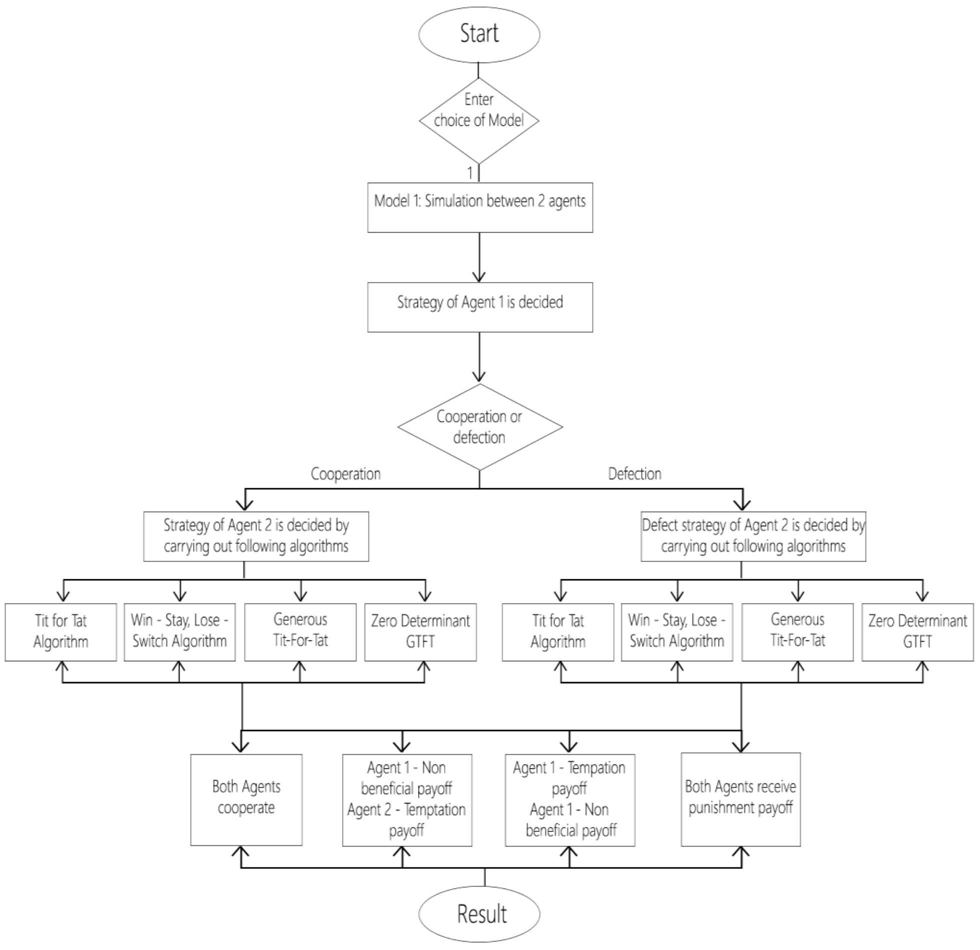

Start
Enter choice of Model
1
Model 1: Simulation between 2 agents
Strategy of Agent 1 is decided
Cooperation or defection
Cooperation
Defection
Strategy of Agent 2 is decided by carrying out following algorithms
Defect strategy of Agent 2 is decided by carrying out following algorithms
Tit for Tat Algorithm
Win - Stay, Lose - Switch Algorithm
Generous Tit-For-Tat
Zero Determinant GTFT
Tit for Tat Algorithm
Win - Stay, Lose - Switch Algorithm
Generous Tit-For-Tat
Zero Determinant GTFT
Both Agents cooperate
Agent 1 - Non beneficial payoff Agent 2 - Temptation payoff
Agent 1 - Tempation payoff Agent 1 - Non beneficial payoff
Both Agents receive punishment payoff
Result

Diagram 1. Model 1 of Case 1 under 5.1.1

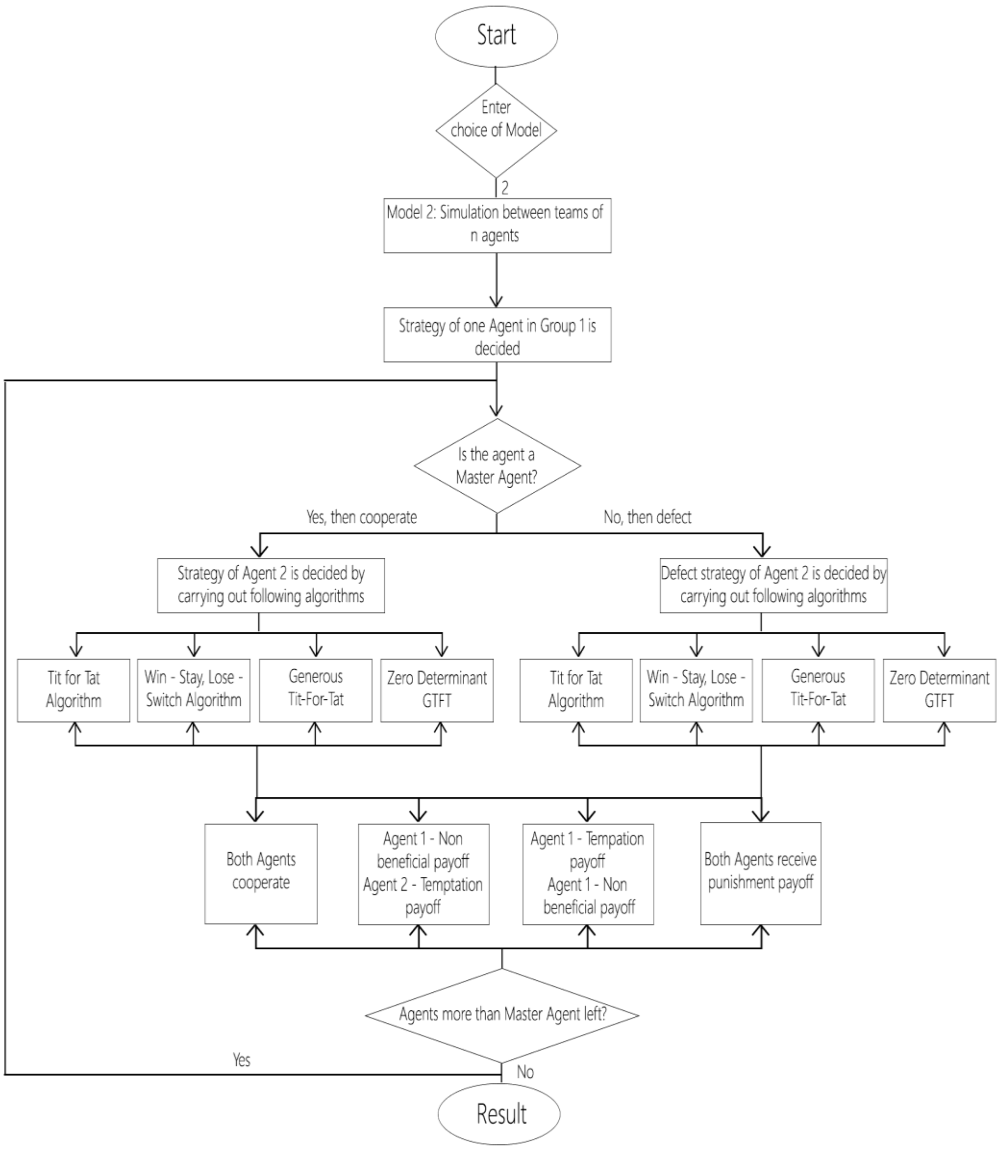

Start
Enter choice of Model
2
Model 2: Simulation between teams of n agents
Strategy of one Agent in Group 1 is decided
Is the agent a Master Agent?
Yes, then cooperate
No, then defect
Strategy of Agent 2 is decided by carrying out following algorithms
Defect strategy of Agent 2 is decided by carrying out following algorithms
Tit for Tat Algorithm
Win - Stay, Lose - Switch Algorithm
Generous Tit-For-Tat
Zero Determinant GTFT
Tit for Tat Algorithm
Win - Stay, Lose - Switch Algorithm
Generous Tit-For-Tat
Zero Determinant GTFT
Both Agents cooperate
Agent 1 - Non beneficial payoff Agent 2 - Temptation payoff
Agent 1 - Tempation payoff Agent 1 - Non beneficial payoff
Both Agents receive punishment payoff
Agents more than Master Agent left?
Yes
No
Result

Diagram 2. Model 2 of Case 2 under 5.1.2

7. List of Strategies/Algorithms [16]

This list consists of those strategies, which already exist. Using these algorithms, we repeatedly carried out execution on the two models and arrived at out most optimized algorithm with efficient runtime.

7.1 TFT (Tit-for-Tat) Strategy

This is an old strategy and still consistently one of the best Prisoner's Dilemma (or any social dilemma game) strategies in existence. The rules are very simple:

- Your first move is always to cooperate.
- You choose what your partner's last choice was.

The advantage with this strategy is that it inevitably evens out to everyone having even T and S outcomes.

7.2 Win-Stay, Lose-Switch (WSLS) Strategy

This strategy was one of the first to "counter" TFT. The rules for this strategy are also very simple:

- Your first move is to cooperate.
- If you encounter a T or R payoff, stay with your previous choice ("win-stay").
- If you encounter a P or S payoff, switch your choice ("lose-switch").

7.3 The Generous Tit-for-Tat (GTFT) Strategy

This is a very simple update to the traditional TFT. The GTFT implements a simple forgiveness factor. GTFT is best when a player has a fuzzy strategy.

- Always cooperate first.
- Choose what your partner chose previously.
- In the case of S payoff, cooperate X% of the time (in most reports, cooperating 10% of the time is enough).

7.4 Zero Determinant GTFT (ZD-GTFT) Strategy

The rules for this strategy are almost identical to the GTFT except for one rule:

- Always cooperate first.
- Choose what your partner chose previously.
- In the case of S payoff, cooperate X% of the time (in most reports, cooperating 10% of the time is enough).
- In the case of T payoff, defect X% of the time (in the perfect ZD-GTFT, you would defect 10% of the time if you are cooperating 10% of the time for S payoffs).

This strategy compensates you for your generosity by being more extortionate for the temptation payoff. For example, say you encounter an S payoff and defect next turn. Your partner chooses to cooperate and you end up with a T payoff. TFT would say to return to a cooperative choice always, with psychological hopes that you are going to return to greater cooperative choices. ZD-GTFT says you can exploit the potentially cooperative environment by "taking some off the top" and defecting again one out of 10 times. Although, you are also generously giving away points at the same rate when the tables are turned. The only difference is that in the case of ZD-GTFT versus GTFT, ZD-GTFT will win out because GTFT is impervious to anything but cooperating if the partner cooperates previously.

7.5 All Defection (ALLD) Strategy

You can imagine the rules for ALLD being probably what they exactly are: Always defect.

This strategy generally comes into play if a player defects initially and finds themselves in a position where they would rather keep what they have (particularly in a T payoff) rather than gain anymore. It only ever works in that specific situation. In games where there is a global variable of competing against not just the one opponent but all other players, ALLD fails because you are up against other strategies with greater payoffs.

7.6 All Cooperation (ALLC) Strategy

The opposite of ALLD: Always cooperate.

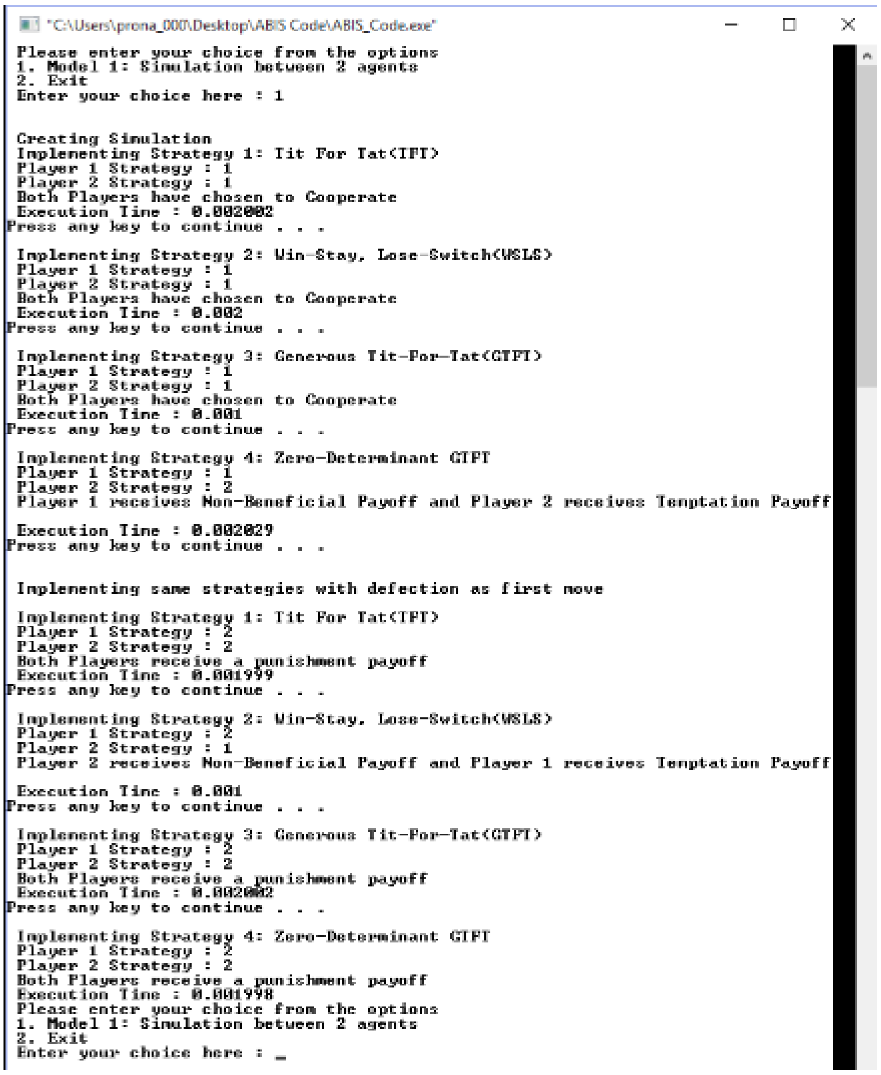
This strategy is chosen if people believe themselves to be on a team as opposed to competing against each other. That notion is preceded by focusing more on the idea of a "global" competition. Another reason why this strategy is sometimes chosen is when the initial choice is a R payoff. Sometimes people believe that R is the most common choice, and therefore choose cooperation to increase that opportunity.

ALLC again fails against almost all other strategies, just like ALLD. Any single defection against ALLC guarantees you winning, regardless of the amount of iterations.

8. Experimentation and Performance Evaluation

8.1 Output

From the performance test on algorithms performed taking into consideration the runtime and most optimized result, we find that when the first move is Cooperation, the Generous Tit-For-Tat Algorithm has the least runtime and most optimized result.



```
"C:\Users\prona_000\Desktop\ABIS Code\ABIS_Code.exe"
 Please enter your choice from the options
 1. Model 1: Simulation between 2 agents
 2. Exit
 Enter your choice here : 1


 Creating Simulation
 Implementing Strategy 1: Tit For Tat(TFT)
 Player 1 Strategy : 1
 Player 2 Strategy : 1
 Both Players have chosen to Cooperate
 Execution Time : 0.002002
Press any key to continue . . .

 Implementing Strategy 2: Win-Stay, Lose-Switch(WSLS)
 Player 1 Strategy : 1
 Player 2 Strategy : 1
 Both Players have chosen to Cooperate
 Execution Time : 0.002
Press any key to continue . . .

 Implementing Strategy 3: Generous Tit-For-Tat(GTFT)
 Player 1 Strategy : 1
 Player 2 Strategy : 1
 Both Players have chosen to Cooperate
 Execution Time : 0.001
Press any key to continue . . .

 Implementing Strategy 4: Zero-Determinant GTFT
 Player 1 Strategy : 1
 Player 2 Strategy : 2
 Player 1 receives Non-Beneficial Payoff and Player 2 receives Temptation Payoff

 Execution Time : 0.002029
Press any key to continue . . .


 Implementing same strategies with defection as first move

 Implementing Strategy 1: Tit For Tat(TFT)
 Player 1 Strategy : 2
 Player 2 Strategy : 2
 Both Players receive a punishment payoff
 Execution Time : 0.001999
Press any key to continue . . .

 Implementing Strategy 2: Win-Stay, Lose-Switch(WSLS)
 Player 1 Strategy : 2
 Player 2 Strategy : 1
 Player 2 receives Non-Beneficial Payoff and Player 1 receives Temptation Payoff

 Execution Time : 0.001
Press any key to continue . . .

 Implementing Strategy 3: Generous Tit-For-Tat(GTFT)
 Player 1 Strategy : 2
 Player 2 Strategy : 2
 Both Players receive a punishment payoff
 Execution Time : 0.002002
Press any key to continue . . .

 Implementing Strategy 4: Zero-Determinant GTFT
 Player 1 Strategy : 2
 Player 2 Strategy : 2
 Both Players receive a punishment payoff
 Execution Time : 0.001998
 Please enter your choice from the options
 1. Model 1: Simulation between 2 agents
 2. Exit
 Enter your choice here : _
```

Figure 4. Sample Output Code with Runtime

When the first move is Defection, the Win-Stay Lose-Switch Algorithm is the most optimized with least runtime as shown in the output analysis given below.

First Move – Cooperation: Win- Stay Lose-Switch Algorithm = 0.002904333s

First Move – Defection: Generous Tit-For-Tat Algorithm = 0.003101533s

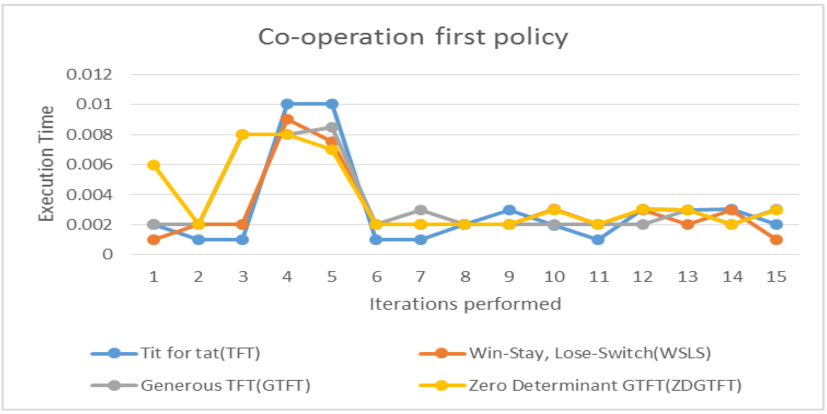
Table 2. Representation of Prisoners' Dilemma

First Move	Algorithm	Runtime
Cooperation	Tit-for-Tat	0.002002s
	Win-Stay Lose-Switch	0.002s
	Generous Tit-for-Tat	0.001s
	Zero-Determinant GTFT	0.002029s
Defection	Tit-for-Tat	0.001999s
	Win-Stay Lose-Switch	0.001s
	Generous Tit-for-Tat	0.002002s
	Zero-Determinant GTFT	0.001998s

8.2 Tables and Graph

Table 3. Output when Always Starting with Co-Operation

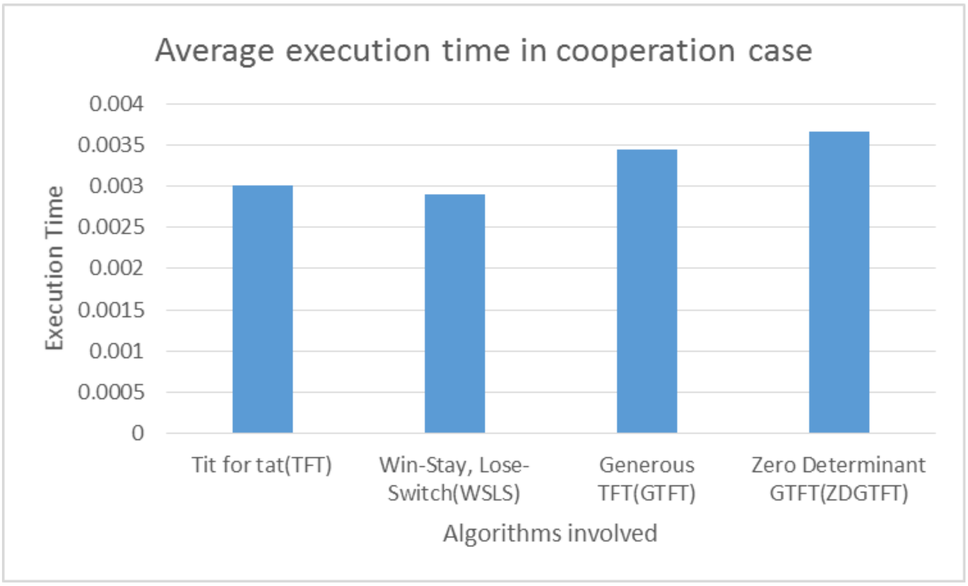
Tit for tat(TFT)	Win-Stay, Lose-Switch (WSLS)	Generous TFT(GTFT)	Zero Determinant GTFT(ZDGTFT)
0.002004	0.001006	0.002001	0.006
0.001002	0.002001	0.001994	0.002
0.001005	0.002003	0.008007	0.008006
0.010014	0.009007	0.008004	0.007999
0.010012	0.007504	0.008505	0.00699
0.000997	0.002	0.002002	0.002007
0.001002	0.002009	0.003002	0.002002
0.002001	0.002001	0.002001	0.002001
0.003001	0.002001	0.002008	0.002001
0.00199	0.003011	0.002	0.002982
0.001	0.002015	0.002017	0.002004
0.003023	0.002996	0.002003	0.003029
0.002998	0.002013	0.003008	0.003005
0.003029	0.002996	0.002002	0.002006
0.002001	0.001002	0.003013	0.002996



Graph 1. Co-Operation First Policy with Execution Time vs Iterations Performed

Table 4. Average Execution Times when Starting with Co-Operation

Tit for tat(TFT)	Win-Stay, Lose-Switch (WSLS)	Generous TFT(GTFT)	Zero Determinant GTFT(ZDGTFT)
0.003005267	0.002904333	0.0034378	0.003668533



Graph 2. Average Execution Time of Algorithms vs. Type of Algorithm Performed in Cooperation Case

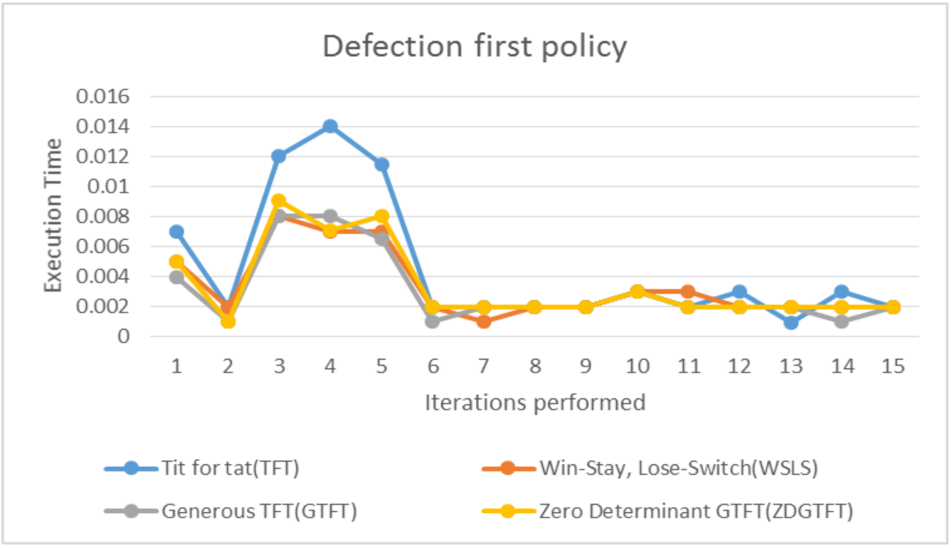
After running a number of iterations of different algorithms in a two-agent environment of the problem considered, we found out that on an average Win-Stay Lose-Switch (WSLS) has the least execution time when the either one of the agent decides to co-operate first.

The trend differs from the results on different iterations but the on an average WSLS Algorithm takes 0.002904333 secs which is the least when compared to other algorithms Win stay lose switch strategies stay with an action if it leads to a satisfactory outcome. Hence, they do not necessarily maximize their payoff. This strategy has been very successful in the context of the iterated Prisoner's Dilemma.

Table 5. Output when Always Starting with Defection

Tit for tat(TFT)	Win-Stay, Lose-Switch (WSLS)	Generous TFT(GTFT)	Zero Determinant GTFT(ZDGTFT)
0.006975	0.005004	0.004004	0.004976
0.001998	0.002002	0.001002	0.000994
0.011994	0.008005	0.008006	0.009051
0.014011	0.007005	0.008006	0.007047
0.011512	0.007	0.006504	0.008006
0.001992	0.002	0.001	0.002
0.002	0.001	0.002007	0.001994
0.002001	0.002005	0.002001	0.002008
0.002	0.002005	0.002007	0.002001
0.003035	0.003003	0.003007	0.00298

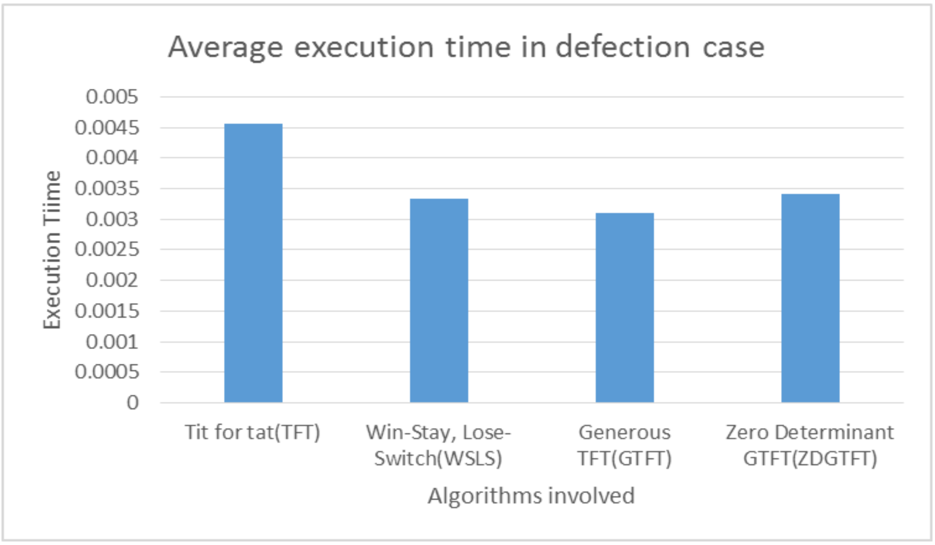
0.001997	0.003008	0.002006	0.002002
0.003009	0.002006	0.001996	0.001996
0.000972	0.001999	0.001996	0.002001
0.002999	0.002001	0.001001	0.002
0.002006	0.002002	0.00198	0.002007



Graph 3. Defection First Policy with Execution Time vs. Iterations Performed

Table 6. Average Execution Times When Starting with Defection

Tit for tat(TFT)	Win-Stay, Lose Switch (WSLS)	Generous TFT(GTFT)	Zero Determinant GTFT(ZDGTFT)
0.004566733	0.003336333	0.003101533	0.0034042



Graph 4. Average Execution Time of Algorithms vs. Type of Algorithm Performed in Defection Case

Considering the second case where one of the agent decides to defect in the first place. After running the existing algorithms discussed above on the second agent, it is found out that on an average Generous tit-for-tat algorithm runs with the least execution time *i.e*. 0.003101533. Based on this algorithm, agent 2 recognizes the move made by agent 1 and decides the move to perform. According to the Generous tit-for-tat algorithm, the agent chooses to imitate the opponent's previous actions. The results of execution times show lot of variations in the initial iterations but gradually becomes constant with in increasing iterations performed.

9. Applications of this Research in Real World Scenarios

1. Environmental Sciences: Prisoner's Dilemma is evident in crisis prediction such as climate changes and natural disaster prediction.
2. Economics: Prisoner's Dilemma is used to predict the profits and losses for co-corporates and their competitors upon implementation of mutual policies. The PD problem is implemented in all decision problems involving trust between two entities.
3. Psychology: The Prisoner's Dilemma problem helps understand various rational approaches to finite choices and how mutual decisions are made.
4. Zoology: To understand the mental capabilities of animals to make mutual choices. Understand the sciences behind partnerships in the lives of different species.

References

[1] R. Axelrod and W. Hamilton, "The evolution of cooperation", Science, vol. 211, **(1981)**, pp. 1390–1396.
[2] F. Herold and Nick Netzer, "Probability Weighting as Evolutionary Second-best", July **(2010)**, pp.32.
[3] "Higher Intelligence Groups have higher cooperation rates in the Repeated Prisoner's Dilemma", Eugenio Proto, Aldo Rustichini, Andis Sofianos, **(2014)**, pp. 2-4.
[4] D. Durai and S. Gratz, "Golden Balls: A Prisoner's Dilemma Experiment" University of Zurich, **(2010)**, pp. 1-5.
[5] H. P. Labs, K. Y. Chen and T. Hogg, "How well do people play a quantum Prisoner's Dilemma?" **(2006)**.
[6] M. Hadzikadic and M. Sun, "A Complex Adaptive System for finding the best strategy for Prisoner's Dilemma", College of Computing and Informatics, University of North Carolina, **(2004)**.
[7] A. J. Stewart and J. B. Plotkin, "Extortion and cooperation in the Prisoner's Dilemma", Department of Biology, University of Pennsylvania.
[8] J. Andreoni and J. H. Miller, "Rational Cooperation in the Finitely Repeated Prisoner's Dilemma: Experimental Evidence", published by Blackwell Publishing for the Royal Economic Society, **(1993)**.
[9] R. Cooper, D. V. D. Jong, R. Forsythe and T. W. Ross, "Cooperation without Reputation: Experimental Evidence from Prisoner's Dilemma Games", **(1992)**.
[10] J. H. Miller, "The coevolution of Automata in the Repeated Prisoner's Dilemma", published by Santa Fe Institute and Carnegie-Mellon University, **(1989)**.
[11] R. Axelrod, "The Evolution of Strategies in the Iterated Prisoner's Dilemma", **(1987)**, pp. 4.
[12] A. Rubinstein, "Finite Automata Play the Repeated Prisoner's Dilemma" published by the Journal of Economic Theory, **(1985)**, pp. 1.
[13] A. Neyman, "Bounded Complexity justifies cooperation in the Finitely Repeated Prisoner's Dilemma," published by Hebrew University of Jerusalem, **(1985)**, pp. 10.
[14] R. Selten and R. Stoecker, "End behaviour in sequences of Finite Prisoner's Dilemma Supergames" published by University of Bonn. **(1985)**, pp. 6-8.
[15] "Replicating different strategies in regards to The Prisoner's Dilemma" published by Nick Wan.
[16] R. Axelrod, "Effective Choice in the Prisoner's Dilemma Problem" published by Sage Publications, **(1980)**, pp. 4.

Authors

Aastha Yadav, She is a student, pursuing a Bachelors of Technology in Computer Science and Engineering at VIT University, Vellore, Tamil Nadu, India. Her research interests include Knowledge based Artificial Intelligence and Network Security. She is a computer science enthusiast and has keen interest in software development and management. She has a particular aptitude to projects with real life application and scope using computer applications.

Chandini Bhambhani, She is a student, pursuing a Bachelors of Technology in Computer Science and Engineering at VIT University, Vellore, Tamil Nadu, India. She is an avid reader and computer software enthusiast. Her interests include Machine Learning under Artificial Intelligence and Data Mining and Warehousing. Chandini has a passion for design and enjoys working with a group of computer enthusiasts as the Vice President of the Computer Society of India, VIT Student Branch, Vellore.

Pronay Peddiraju, He is a student pursuing his Bachelors of Technology in Computer Science and Engineering at the Vellore Institute of Technology, Vellore, TN, India. His accomplishments involve projects in the fields of Game Development, 3D Modeling and Simulation, Content Creation, Microcontrollers, Enthusiast Grade Computer Hardware and C++ Programming. He is involved as an active member of the Creation Labs, VIT Vellore as a 3D Asset Developer for the Game Development Team and is currently the Project Director for the Computer Society of India – Student Chapter in VIT University. He is also very involved in development fests and hackathons by implementing his skills in 3D prototyping and development.

Ronnie D. Caytiles, He had his Bachelor of Science in Computer Engineering- Western Institute of Technology, Iloilo City, Philippines, and Master of Science in Computer Science– Central Philippine University, Iloilo City, Philippines. He finished his Ph.D. in Multimedia Engineering, Hannam University, Daejeon, Korea. Currently, he serves as an Assistant Professor at Multimedia Engineering department, Hannam University, Daejeon, Korea. His research interests include Mobile Computing, Multimedia Communication, Information Technology Security, Ubiquitous Computing, Control and Automation.

N. Ch. S. N. Iyengar, He is a Professor of the School of Computer Sciences and Engineering at VIT University, Vellore, TN, India. His research interests include Distributed Computing, Information Security, Intelligent Computing, and Fluid Dynamics (Porous Media). He has had teaching and research experience with a good number of publications in reputed International Journals & Conferences. He chaired many International Conferences delivered Keynote lectures, served as PC Member/Reviewer. He is an Editorial Board member for many International Journals like Int. J.

of Advances in Science and Technology, of SERSC, Cybernetics and Information Technologies (CIT)-Bulgaria, Egyptian Computer Science Journal-Egypt, IJConvC of Inderscience-China, IJCA (USA) *etc.*, Also Editor in Chief for International Journal of Software Engineering and Applications(IJSEA) of AIRCC, Advances in Computer Science (ASC) of PPH, Guest editor for "Cloud Computing and Services" IJCNS.