KanGAL Report Number 2006002

Optimal Strategies of the Iterated Prisoner's Dilemma Problem for Multiple Conflicting Objectives

Shashi Mittal
Dept. of Computer Science and Engineering
Indian Institute of Technology, Kanpur, India
mshashi@iitk.ac.in

Kalyanmoy Deb
Dept. of Mechanical Engineering
Indian Institute of Technology, Kanpur, India
deb@iitk.ac.in

***Abstract*— In this paper, we present a new paradigm of searching optimal strategies in the game of Iterated Prisoner's Dilemma using multiple objective evolutionary algorithms. This method is better than the existing approaches, because it not only gives strategies which perform better in the iterated game, but also gives a family of non-dominated strategies, which can be analyzed to see what properties a strategy should have to win in the game. We present the results obtained with this new method, and also the common pattern emerging from the set of non-dominated strategies so obtained.**

Keywords: Games, Prisoner's dilemma, Strategies, Evolutionary algorithms

I. Introduction

The prisoner's dilemma is a well known game that has been extensively studied in economics, political science, machine learning [1], [2] and evolutionary biology [3]. In this game, there are two players, each of whom can make one of the two moves available to them Cooperate (C) or Defect (D). Both players choose their moves simultaneously and independent to each other. Depending upon the moves chosen by either player, each of them gets some payoff. The payoff matrix is shown in Figure1.

		Player 2	
	Decision	Cooperate	Defect
Player 1	Cooperate	R=3 R=3	S=0 T=5
	Defect	T=5 S=0	P=1 P=1

R: REWARD S: SUCKER T: TEMPTATION P: PENALTY

Fig. 1. The classical choice for payoff in Prisoner's Dilemma (Player 1's payoffs are given first).

When both players cooperate, they are awarded at an equal but intermediate level (the reward, R). When only one player defects, he receives the highest possible payoff (the temptation, T) while the other player gets the sucker's payoff (the sucker, S). When both the players defect, they receive and intermediate penalty (the penalty, P).

Several interesting properties of the game can be immediately observed. It can be seen that this is a non-zero sum game (that is, the sum of the payoffs of the two players is not always a constant), and hence there is no single universal strategy which will work for all game plays for a player. In a one-shot game, both the players will choose to Defect (D, D), because this move is guaranteed to maximize the payoff of the player no matter what his opponent chooses. However, it can be seen that both players would have been better off choosing to cooperate with each other (hence the dilemma).

In game theory, the move (D, D) of the players is termed as a *Nash Equilibrium* [4], which is a steady state of the game in which no player has an incentive to shift from its strategy. Nash [5] proved that any n-player game has a Nash Equilibrium, when randomization in choosing the moves is permitted. However, as it is clear from the prisoner's dilemma game, a Nash Equilibrium may not necessarily be the social optimum.

The situation becomes more interesting when the players play this game iteratively (called the Iterated Prisoner's Dilemma or IPD) and the payoffs are accumulated over each iteration. If both the players have no idea about the number of iterations beforehand, then it is possible to have an equilibrium which is better than (D, D). The equilibrium outcomes in iterated games are defined by folk theorems [6]. For prisoner's dilemma, there are infinitely many equilibrium outcomes, in particular it is possible to have an equilibrium outcome in which both the players always cooperate.

Suppose that there are a number of players, and each player players the iterated game with other players in a round robin fashion, the scores being cumulated over all the games. The winner of the game is the player with the maximum payoff at the end of the round robin tournament. The problem that we consider in this paper is to find optimal strategies which will ensure victory in such a tournament. This has been a widely studied problem by game theorists and artificial intelligence experts alike. Axelrod was the first to study this problem in detail [7], [1], [8]. He used single-objective evolutionary

algorithm for finding the optimal strategies. This is discussed in section 2. Since Axelrod, there have been several studies on this problem [9], [10], [11], [12], [13].

However, in all these studies, the problem of finding optimal strategies has been viewed as a single-objective problem. That is, the objective is to find strategies which maximize their own score in a round robin tournament. In this paper, we present a new approach of finding optimal strategies by considering the problem as a multiple objective optimization problem: maximizing self-score and minimizing opponent score. Such an approach has not been previously investigated in literature before. We discuss this approach in detail in section 3, the details of the simulations performed in section 4 and the results obtained in section 5.

II. Axelrod's study

Axelrod organized two tournaments in the year 1985 and invited strategies from a number of experts and game theorists. To his surprise, he found that the winner in both the tournaments was a very simple strategy, namely 'Tit for Tat'. This strategy cooperates on the first move, and then simply copies the opponent's last move in it's subsequent move. That such a simple strategy turned out to be the winner was quite surprising, and Axelrod set out to find other simple strategies with the same or greater power. Axelrod adopted a simple but elegant way for encoding strategies [1], and then used single-objective evolutionary algorithm to obtain optimal strategies. The encoding scheme is described in detail here.

For each move in the game, there are four possibilities: both the players can cooperate (CC or R for reward), the other player can defect (CD or S for sucker), the first player can defect (DC or T for temptation), or both the players can defect (DD or P for penalty). To code the particular strategy, the particular behavioral sequence is coded as a three letter string. For example, RRR would represent the sequence where both the players cooperated over the previous three moves and SSP would represent the sequence where the first player was played for a sucker twice, and then finally defected. This three letter sequence is then used to generate a number between 0 and 63, by interpreting it as a number in base 4. One such possible way is to assign a digit value to each of the characters in following way: $CC = R = 0$, $DC = T = 1$, $CD = S = 2$ and $DD = P = 3$. In this way, RRR would decode to 0, and SSP will decode to 43. Using this scheme, a particular strategy can be defined as a 64-bit binary string of C's (cooperate) and D's (defect) where the ith C or D corresponds to the ith behavioral sequence. Figure 2 shows such an example GA string. For the example string in the figure, the three-letter code comes to be RTR for the previous moves (given in the figure). This decodes to 4, thereby meaning that player 1 should play the (4+1) or 5-th move specified in the first 64-bit GA string. In this case, the fifth bit is C, meaning that the player 1 will cooperate.

Since a particular move depends on the previous three moves, so the first three moves in a game are undefined in the above scheme. To account for these six bits (C's and D's, initially assigned at random) are appended to the above 64 bit string to specify a strategy's *premises*, or assumption about the pre-game behavior. Together, each of the 70 bit strings thus represent a particular strategy, the first 64 for rules and the next 6 for the premises.

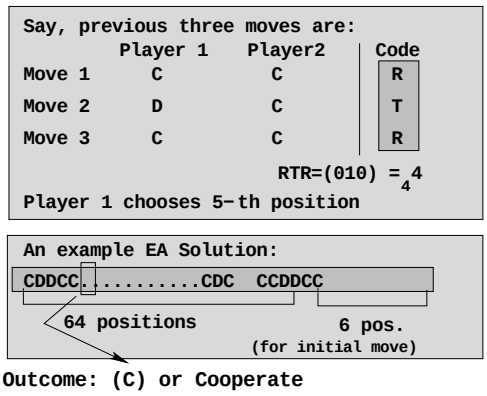

Fig. 2. Encoding a strategy for IPD.

Axelrod used the above encoding scheme to find optimal strategies using a single-objective genetic algorithm. He found that from a random start, the genetic algorithm discovered strategies that not only performed quite well, but also beat the overall performance of 'Tit for Tat' strategy, mentioned earlier.

In this work, the encoding scheme is the same as that mentioned above. However, in addition to a single-objective EA, we use a MOEA to optimize the strategy. The two objectives chosen are: (i) maximizing the self-score and (ii) minimizing the opponent's score. Here the opponent's score means the cumulative score the opponents scored when playing against a particular strategy.

III. Using multiple objective evolutionary algorithms

Most studies of IPD considered a single-objective of maximizing a player's own score. In this paper, for the first time, we treat the problem as a bi-objective optimization problem of maximizing the player's own score and simultaneously minimize opponent's score.

A. Why multiple objective evolutionary algorithm?

The original formulation of the prisoner's dilemma game looks like a single-objective problem, that is, to find a strategy which maximizes a player's self-score. However, this problem can also be looked as a multiple objective optimization problem. It is possible to win the game by not only maximizing the self-score, but also by minimizing the opponent's score. Since the prisoner's dilemma game is a non-zero sum game, it is possible that there is a trade-off between these two objectives (we will later show that this is actually the case), and therefore using a multiple objective evolutionary algorithm may actually give a better insight to the optimal strategies of playing the

game as compared to a single-objective formulation. This is because using multiple conflicting objectives, not one but a number of trade-off optimal solutions can be found. These non-dominated trade-off solutions so obtained can then be analyzed to look for any pattern or insights about optimal strategies for the IPD. If any such patterns are discovered, they would provide a blue-print in formulating optimal strategies for the game.

B. The NSGA-II algorithm

For multiple objective optimization, we use the NSGA-II algorithm given by Deb et al. [14]. NSGA-II has been successfully applied to many other multiple objective optimization problems [15] as well.

IV. Simulations and test cases

Both single-objective EA and MOEA were used for getting optimal strategies. The simulation for both the algorithms followed these steps. In each generation, a certain number of strategies were generated, and each strategy was made to play against 16 other players. Each game consisted of 150 moves. Then the strategies were sorted in the decreasing order of their cumulative scores, and the next generation was created using a recombination operator. The payoff matrix was the same as shown in Figure 1. The details of 16 other players in the fray have been given in the appendix. These strategies have been used extensively in previous studies on IPD.

Clearly, in one particular game, a player can score a maximum of 750 (if he always defects, and the opponent always cooperates), and a minimum of 0 (if he always cooperates, while the opponent always cooperates). None of these two extremes are achieved in practice. According to Dawkins [3], a more useful measure of a strategy is how close it comes to the *benchmark score*, which is the score a player will have if both the players always cooperate. In this case, the benchmark score is 450. For example if the score of a player, averaged over all the players he played, is 400, then he has scored 89% of the benchmark score. This is a more useful way of denoting the score of a player, since it is independent of the particular payoff matrix used, as well as the number of players against which the player played. In all the results presented in the next section, we will refer only to the average score of a player in a game, or the score as a percentage of the benchmark score.

V. Simulation Results

Now, we present and analyze the results obtained by NSGA-II.

A. Results obtained using single-objective EA

The single-objective EA used is the same as used by Axelrod [1]. Two runs of single-objective EA were done. One, in which the self-score of the player was maximized, and the other in which the opponent's score was minimized. For each of the runs, the population size was fixed at 40. The results obtained when the EA is run for 200 generations are shown in Figure 3 and 4.

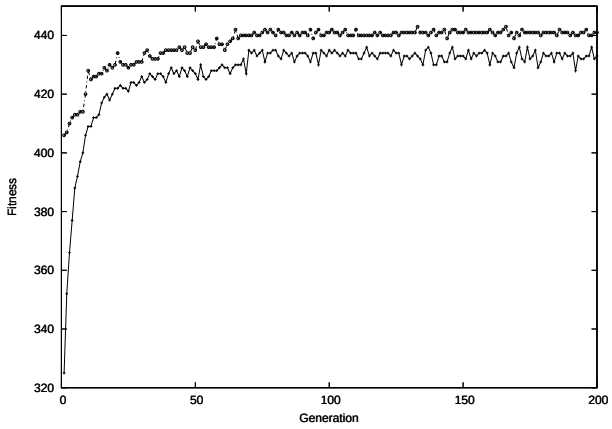

Fig. 3. Plot of the mean fitness (shown in solid line) and maximum fitness (shown in dotted line) of population when self-score is maximized.

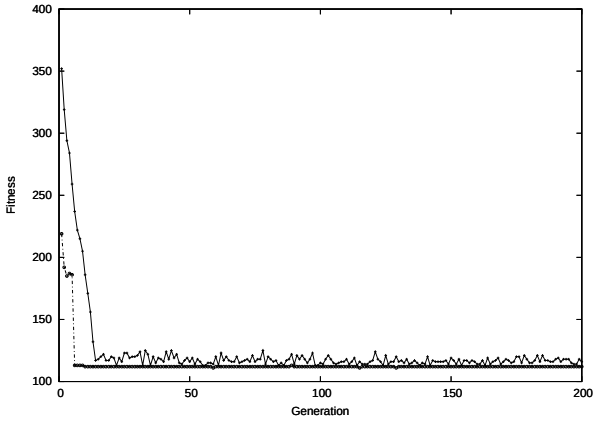

Fig. 4. Plot of the mean fitness (shown in solid line) and minimum fitness (shown in dotted line line) of population when opponent score is minimized.

For maximizing the self-score, the fitness measure of a sample is its self-score, hence the fitness score is to be maximized, while in the second score the fitness is the opponent score (score the opponent had when playing against this player), which is minimized.

As is clear from the graphs, in the first case the mean fitness increases steadily, and after 200 generations the maximum self-score of all of a sample in the population is 441, which is 97% of the benchmark score. When the EA is run for longer generations, the maximum fitness converges at 442 and does not increase further. When these optimal strategies are fielded in a round robin tournament, these strategies win with a big margin. Tables 1 and 2 show the outcome (average of 20 runs) of two tournaments. In the first tournament, there are 16 strategies and 'Tit for Tat' is the winner with an average score of 387. In the second tournament, when the single-objective optimal strategy is fielded, it wins by a huge margin, scoring as high as upto 97% of the benchmark score. This is in line with the results obtained by Axelrod. We refer to the strategy obtained by maximizing the self-score as "Strategy SO".

When the opponent score is minimized, the minimum fitness stabilizes at 112. The strategies so obtained perform poorly in a round robin tournament (their performance is quite similar to that of the Always Defect strategy). We refer to this strategy as "Strategy SO-min". Table 3 shows the average score of the players when this strategy is included in the tournament. It can be seen that this strategy performs as bad as the Always Defect strategy. As such, it seems that there is little incentive in minimizing the opponent score. However, this is not the case, as the results of the next subsection will show.

TABLE I
TOURNAMENT 1.

Player	Average score
Tit for Tat	387
Soft Majority	379
Tit for two tats	379
Spiteful	376
Hard Tit For Tat	370
Always Cooperate	359
Periodic Player CCD	354
Naive Prober	353
Pavlov	351
Periodic Player CD	351
Remorseful Prober	351
Random Player	323
Hard Majority	317
Suspicious Tit for Tat	310
Periodic Player DDC	309
Always Defect	305

TABLE II
TOURNAMENT 2.

Player	Average score
Strategy SO	438
Tit for Tat	390
Soft Majority	384
Tit for two tats	384
Spiteful	381
Hard Tit For Tat	374
Always Cooperate	364
Naive Prober	359
Remorseful Prober	357
Pavlov	357
Periodic Player CCD	336
Periodic Player CD	335
Suspicious Tit for Tat	319
Hard Majority	312
Random Player	310
Periodic Player DDC	296
Always Defect	296

B. Results of MOEA

The parameters used in MOEA are as follows: size of the population = 200, and the algorithm was run for 200 generations. NSGA-II algorithm [14] was used.

1) Evolution of optimal strategies: Starting from a purely random distribution, the strategies ultimately converged to the Pareto-optimal front. This is shown in Figure 5. It shows that the MOEA is indeed successfully able to search the solution space for optimal results. It also shows that there is a trade-off between maximizing the self-score and minimizing the opponent's score. In Figure 6, the Pareto-optimal fronts with a single-objective EA and a few other strategies are shown, when both the single-objective EA as well as the MOEA is run for 20,000 generations. After using NSGA-II to obtain the non-dominated front, a local search was performed from each of the member of this front, and it was found that there was little or no improvement in the solution. Therefore, the non-dominated front obtained using NSGA-II is indeed very close to the actual Pareto-optimal front. The use of local search from MOEA solutions ensures that the final solution is at least locally optimal.

TABLE III
TOURNAMENT 3.

Player	Average score
Tit for Tat	372
Soft Majority	375
Tit for two tats	365
Spiteful	363
Hard Tit For Tat	356
Naive Prober	341
Always Cooperate	337
Remorseful Prober	336
Periodic Player CCD	335
Periodic Player CD	334
Pavlov	344
Random Player	309
Hard Majority	306
Suspicious Tit for Tat	300
Always Defect	296
Strategy SO-min	296
Periodic Player DDC	296

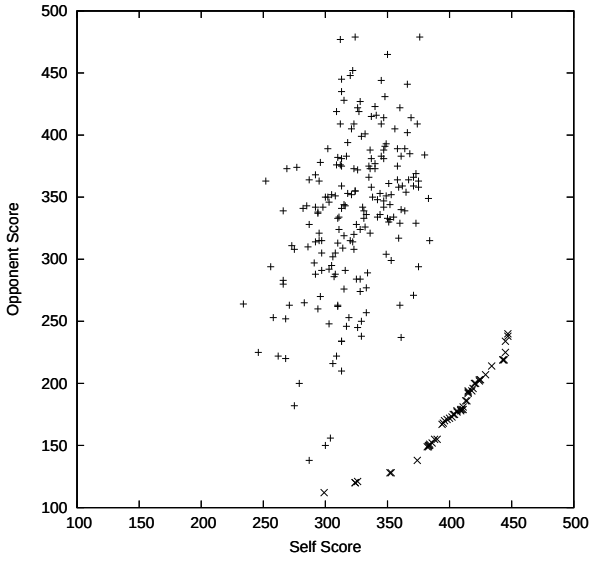

Fig. 5. The initial random solution (shown with '+') and the non-dominated front(shown in 'x'), when NSGA-II is run for 20000 generations.

The most significant outcome of the MOEA, is however, evolution of strategies which perform much better that those

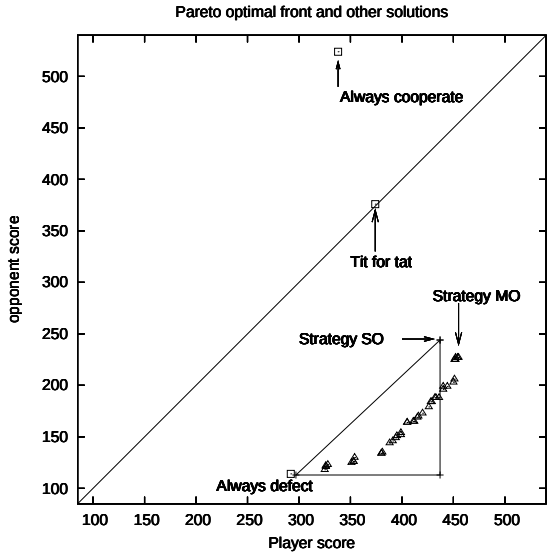

Fig. 6. The non-dominated solutions, together with the single objective EA results (the upper and the left vertexes of the triangle) and a few other strategies.

obtained using earlier methods. The strategy with the maximum self-score (the maximum score, 451 is slightly better than that for the optimal strategy obtained using single-objective EA, 442) had a mean opponent score (214) that was significantly lower than that for the single-objective optimal strategy (244). Figure 6 shows the single-objective optimum strategy (Strategy SO) and the multiple objective optimum strategy (Strategy MO). Strategy MO so obtained not only outperformed other strategies in a round robin tournament (see Table 4 and Table 5), but also defeated the Strategy SO (Table 5). This clearly shows that MOEA is able to find better strategies as compared to the single-objective EA.

TABLE IV
TOURNAMENT 4.

Player	Average score
Strategy MO	448
Tit for Tat	391
Hard Tit For Tat	375
Soft Majority	370
Tit for two tats	370
Spiteful	363
Naive Prober	358
Remorseful Prober	344
Always Cooperate	337
Periodic Player CCD	336
Periodic Player CD	334
Pavlov	334
Suspicious Tit for Tat	319
Hard Majority	312
Random Player	310
Periodic Player DDC	296
Always Defect	296

The other extreme solution on the Pareto-optimal front is the same as obtained by minimizing the opponent's score, and has the same performance as the Always Defect strategy. Even though a many of the bit-positions in the strategy string for this strategy are C, it behaves almost like the Always Defect strategy, as is discussed later.

TABLE V
TOURNAMENT 5.

Player	Average score
Strategy MO	431
Strategy SO	421
Tit for Tat	394
Hard Tit For Tat	379
Soft Majority	375
Tit for two tats	374
Spiteful	368
Naive Prober	362
Remorseful Prober	351
Always Cooperate	343
Pavlov	341
Suspicious Tit for Tat	327
Periodic Player CD	320
Periodic Player CCD	320
Hard Majority	307
Random Player	296
Always Defect	288
Periodic Player DDC	286

2) Relationship among the Pareto-optimal strategies: The fact that a non-dominated front is obtained by using MOEA indicates that the strategies lying on this front must have something in common. As such, different Pareto-optimal strategies look quite different from each other. To have a closer look at these strategies, during the game, the number of times each bit position in the string was used in a round-robin tournament was recorded, and plotted for different strategies. Figure 7 shows the combined plot for six Pareto-optimal strategies (chosen from Figure 6), and for six random strategies (for comparison).

In the plot, the frequency distribution for six Pareto-optimal strategies are given in the lower half (with self-score decreasing along the y-axis), and for six randomly chosen random strings in the upper half. The cooperative moves are shown in white boxes, and the defecting moves are shown in black boxes. Only those bit positions which were used more than 20 times in the round-robin tournament are shown. The plot reveals that only a few of the bit positions of a strategy are used more than 20 times. Also, the Pareto-optimal strategies show some interesting similarities with respect to the usage of a particular bit position. For example, positions 0, 1, 4, 5, 17, 18, 21, and 55 turn out to be 'Defecting' in all of the six Pareto-optimal solutions. There are also some trends in 'Cooperating', coming out as common strategies of these high-performing solutions. We discuss a few of them in the following:

- **0** : This decodes to *PPP*, i.e. both the players have been defecting with each other over the previous three moves. Since both players are defecting, it is expected that the player 1 should also defect as a good strategy for preventing the opponent's score to be high in the subsequent moves.

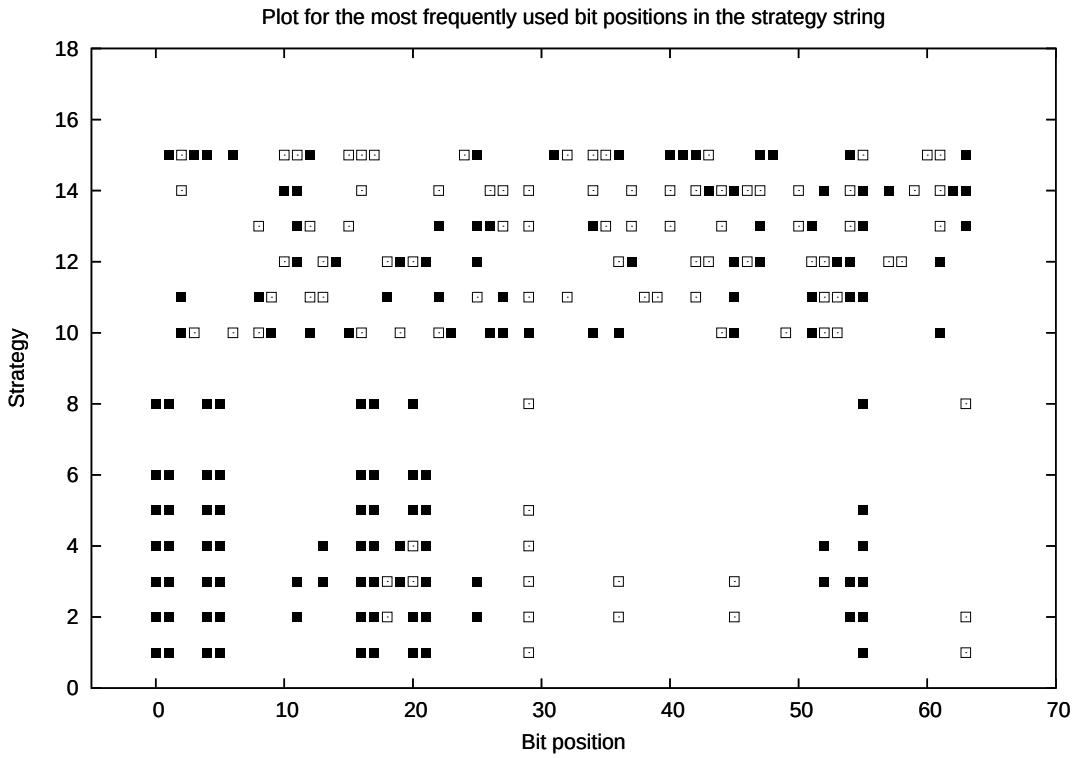

Fig. 7. Plot of the frequently used bit positions in the strategy strings.

- **1** : This decodes to *PPT*. The opponent defected on the first two moves, but did not do so in the third move, while player 1 defected in all the three moves. In this case, the strategy is to defect, so as to "exploit" the foolish opponent.
- **4** and **5** : decodes to *PTP* and *PTT*, which are similar to the previous case, and again the moves are to defect to exploit the opponent.
- **17** : This implies *TPT*, i.e. player 1 defected on all the previous three moves, but the opponent was foolish enough to cooperate, clearly in this case, player 1 will defect, to exploit the opponent.
- **29** : 29 decodes to *STR*. This represents "reconciliation", that is, both the players initially defected, but cooperated on the last move. So the two players are trying to ensure cooperation, and hence the next move in a good game-playing strategy would be to cooperate.
- **55** : 55 stands for *RTR*. This again a case of exploitation, since the opponent cooperated on all the previous three moves even though I defected once. Hence the move in this situation is to defect.
- **63** : 63 is *RRR*, that is the players cooperated on all the previous three moves. Since both the players are cooperating, so the best move in this case is to continue cooperating.

The eighth solution in the figure is for the single-objective optimum strategy of maximizing self-score alone. Interestingly, the frequently used moves in this strategy is similar to the 1st strategy of the bi-objective Pareto-optimal solutions (with the highest self-score). Thus, a recipe for maximizing self-score by minimizing the opponent's score is to learn to defect when the opponent is either defecting or foolishly trying to cooperate when player 1 is continuously defecting. Another recipe to follow is to cooperate with the opponent when the opponent has indicated its willingness to cooperate in the past moves.

Another matter to note is that as the self-score decreases (as solutions go up on the y axis), the strategies become more and more defecting and the frequency of cooperation reduces. To minimize the opponent's score, the payoff matrix indicates that player 1 should defect more often. When this happens, self-score is also expected to be low against intelligent players, because both players will engage in defecting more often.

For random strategies, no such pattern is observed. It can be seen that for the Pareto-optimal strategies, most of the bit positions are either sparingly used, or are not used at all. For example, the strategy with the least self-score always makes defecting moves, even though there are many C's in its strategy string, showing that it behaves almost like the Always Defect strategy.

VI. Significance of the results

The above results demonstrate the power of MOEAs in searching better strategies for the IPD problem. The optimal

strategies obtained using this method outperforms all other strategies. In particular, it performs better than the strategies obtained using single-objective optimization procedure. This shows that MOEAs are a more useful methods for finding optimal strategies, as compared to single-objective EAs.

The fact that a non-dominated front (which is quite close to the actual Pareto-optimal front) shows that there is indeed a trade-off between maximizing self-score and minimizing opponent score. Therefore, to be successful in a round robin tournament, a player should not only try to maximize its own score, but also minimize the opponent score.

We further observe that the strategies lying on the non-dominated front share some common properties. This can give us valuable insight about the optimal strategies for a round robin tournament. It will be interesting to make some prototype strategies using these common features and observing their performance in a tournament; we leave this for a future research work.

We had also carried out another simulation in which we used NSGA-II to minimize two other objectives: Maximize self-score, and minimize the maximum of the opponent scores (in previous case, we had minimized the average score of the opponents). The Pareto-optimal front obtained is shown in Figure 8 by marking the obtained strategies (maximum opponent score) with 'X'. When the average score (marked

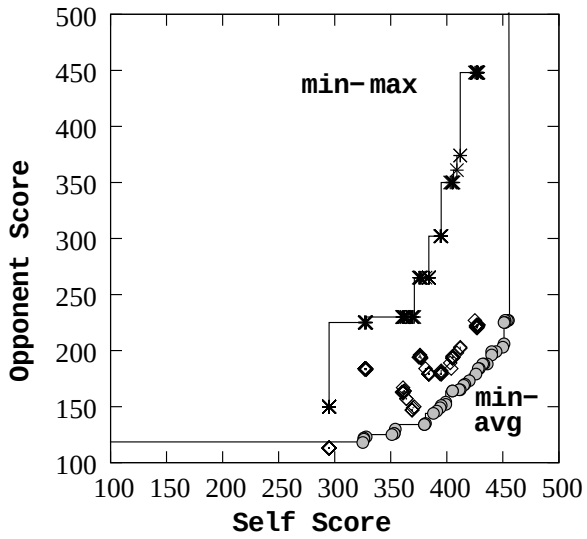

Fig. 8. Pareto-optimal front obtained when maximum of the opponent score is minimized (maximum opponent score is represented by 'X' and corresponding average score is represented by diamonds) against the Pareto-optimal front obtained earlier (shown in circles).

with diamonds) over all opponent players are computed and plotted for these strategies, they are found to be dominated by the previous Pareto-optimal solutions. The second objective value of Pareto-optimal solutions using this method is worse than before, as the maximum of the opponents' scores is going to be always more than average of opponents' scores. A good spread in solutions is still obtained, but since this new objective is non-differentiable, the obtained front in this case is not as smooth as before. As discussed above, the solutions obtained are also inferior from the earlier solutions. Based on this study, we may conclude that minimizing the average score of opponents is a better optimization strategy than minimizing the maximum score of opponents.

VII. Conclusions

We have presented a new paradigm for searching optimal strategies in Iterated Prisoner's Dilemma (IPD) using a multi-objective optimization procedure. Such a method has not been used before in the literature for this problem. It has been revealed that such a solution strategy has several advantages over the existing single-objective methods for finding useful optimal game-playing strategies. Hopefully, such an approach will find further application in related game-playing problems in the near future.

References

[1] R. Axelrod, "The evolution of strategies in the iterated prisoner's dilemma," in *Genetic Algorithms and Simulated Annealing*, L. Davis, Ed. Los Altos, CA: Morgan Kaufmann, 1987.

[2] D. E. Goldberg, *Genetic Algorithms for Search, Optimization and Machine Learning*. Reading: Addison-Wesley, 1989.

[3] R. Dawkins, *The Selfish Gene*. New York: Oxford University Press, 1989.

[4] M. Rubenstein and M. Osborne, *A Course in Game Theory*. MIT Press, 1994.

[5] J. F. Nash, "Equilibrium points in n-person games," in *Proceedings of the National Academy of Sciences*, vol. 36, 1950, pp. 48–49.

[6] D. Fudenberg and E. Maskin, "The folk theorem in repeated games with discounting or incomplete information," *Econometrica*, vol. 54, no. 3, 1986.

[7] R. Axelrod and W. Hamilton, "The evolution of coperation," *Science*, vol. 211, pp. 1390–6, 1981.

[8] R. Axelrod, *The Evolution of Cooperation*. New York: Basic Books, 1989.

[9] D. Fogel, "Evolving behaviors in the iterated prisoner's dilemma," *Evolutionary Computation*, vol. 1, pp. 77–97, 1983.

[10] M. Nowak and K. Sigmund, "A strategy of win-stay, lose-shift that outperforms tit-for-tat in the prisoner's dilemma game," *Nature*, vol. 364, pp. 56–58, 1993.

[11] B. Beaufils, J. P. Delahaye, and P. Mathieu, "Our meeting with gradual, a good strategy for the iterated prisoner's dilemma," in *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, C. Langton and K. Shimohara, Eds. Cambridge, MA, USA: The MIT Press, 1996, pp. 202–209.

[12] D. Bragt, C. Kemenade, and H. Poutr, "The influence of evolutionary selection schemes on the iterated prisoner' s dilemma," *Computational Economics*, vol. 17, pp. 253–263, 2001.

[13] D. Jang, P. Whigham, and G. Dick, "On evolving fixed pattern strategies for iterated prisoner's dilemma," in *Proceedings of the 27th conference on Australasian computer science*, Dunedin, New Zealand, 2004, pp. 241–247.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[15] K. Deb, *Multiobjective Optimisation Using Evolutionary Algorithms*. Chichester, U.K.: Wiley, 2001.

Appendix I

Details about the different strategies used in the round-robin tournament:

1) **Always Cooperate** : Cooperates on every move
2) **Always Defect** : Defects on every move

3) **Tit for Tat** : Cooperates on the first move, then simply copies the opponent's last move.
4) **Suspicious Tit for Tat** : Same as Tit for Tat, except that it defects on the first move
5) **Pavlov** : Cooperates on the first move, and defects only if both the players did not agree on the previous move.
6) **Spiteful** : Cooperates, until the opponent defects, and thereafter always defects.
7) **Random Player** : Makes a random move.
8) **Periodic player CD** : Plays C, D periodically.
9) **Periodic player DDC** : Plays D, D, C periodically.
10) **Periodic player CCD** : Plays C, C, D periodically.
11) **Tit for Two Tats** : Cooperates on the first move, and defects only when the opponent defects two times.
12) **Soft Majority** : Begins by cooperating, and cooperates as long as the number of times the opponent has cooperated is greater than or equal to the number of times it has defected, else it defects.
13) **Hard Majority** : Defects on the first move, and defects if the number of defections of the opponent is greater than or equal to the number of times it has cooperated, else cooperates.
14) **Hard Tit for Tat** : Cooperates on the first move, and defects if the opponent has defects on any of the previous three moves, else cooperates.
15) **Naive Prober** : Like Tit for Tat, but occasionally defects.
16) **Remorseful Prober** : Like Naive Prober, but it tries to break the series of mutual defections after defecting.