2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

TURAN: Evolving Non-Deterministic Players For The Iterated Prisoner's Dilemma

M. Gaudesi, E. Piccolo, G. Squillero
DAUIN, Politecnico di Torino
Torino, Italy
Email: {marco.gaudesi,elio.piccolo,giovanni.squillero}@polito.it

A. Tonda
UMR 782 GMPA, INRA
Thiverval-Grignon, France
Email: alberto.tonda@grignon.inra.fr

***Abstract*—The *iterated prisoner's dilemma* is a widely known model in game theory, fundamental to many theories of cooperation and trust among self-interested beings. There are many works in literature about developing efficient strategies for this problem, both inside and outside the machine learning community. This paper shift the focus from finding a "good strategy" in absolute terms, to dynamically adapting and optimizing the strategy against the current opponent. *Turan* evolves competitive non-deterministic models of the current opponent, and exploit them to predict its moves and maximize the payoff as the game develops. Experimental results show that the proposed approach is able to obtain good performances against different kind of opponent, whether their strategies can or cannot be implemented as finite state machines.**

I. Introduction

The *prisoner's dilemma* is a well-known game-theory model, describing a simple situation that offers different rewards for a selfless and a selfish behavior, and where the outcome is determined by both choices. It was originally conceived by Merrill Flood and Melvin Dresher in 1950 as part of the RAND Corporation[1] investigations into game theory and its possible applications to global nuclear strategy [1]. A few years later, in order to make the idea more accessible to psychologists, Albert Tucker depicted two criminals arrested for an offense and placed in separate isolation cells. Each villain could opt to cooperate with his accomplice by negating any involvement in the crime, or betray the (former) partner by confessing to the police [2], hence the name.

Intuitively, and disregarding moral considerations, in a single turn the more convenient behavior is always the selfish one. If one player is selfless, the most profitable action for the other one is to exploit the good faith being selfish. On the other hand, if one player is selfish, the best action for the opponent is, again, being selfish too, minimizing the damage. Hence, the situation has no strategic interest.

In the iterated prisoner's dilemma (IPD), however, the two players compete for an unknown, potentially infinite, number of rounds, with a memory of the previous exchanges. Noticeably, the above *defection strategy* fails badly to predict the behavior of human players, leaving the field open to many possibly viable alternatives. In particular, Aumann showed that rational players interacting for indefinitely long games can sustain a fully cooperative outcome [3]. While extremely simple, the IPD is used in several fields to describe cooperation and trust, and research on the topic is still thriving, with important contributions regularly appearing in literature.

In 1979, Axelrod organized the first important IPD's tournament, soliciting strategies from game theorists who had published in the field [4]. The 14 entries were competed along with a fifteenth one which cooperates or defects with equal probability. Each strategy was played against all others over a sequence of 200 moves. The winner of the tournament was submitted by Anatol Rapoport, a Russian-born American mathematical psychologist. His strategy was named *Tit for Tat*: cooperate on the first move, and then mimic whatever the other player did on the previous move. In a second tournament, Axelrod collected 62 entries and, again, the winner was *Tit for Tat*. Indeed, the tit-for-tat strategy is robust because it never defects first and is never taken advantage of for more than one iteration at a time.

Since then, new tournaments have been held regularly, with sometimes counter-intuitive or baffling results. For example, in 2004 the Southampton School of Electronics and Computer Science gained the first three positions by sending in the tournament over 60 strategies that were able to recognize each other with an initial 10-move handshake and then *collude*, raising the score of selected members of their own *team* while lowering the score of other players [2].

Evolutionary algorithms (EAs) have been employed to create competitive players since Axelrod's initial experiments [5]. A comprehensive survey of recent attempts to find efficient strategies is reported in [6]. Fogel [7] uses *evolutionary programming* to investigate the conditions that promote cooperative behavior. Franken and Engelbrecht [8] present a *particle swarm optimization* technique to evolve competitive strategies. Gohneim et al. used evolutionary computation to efficiently allocate memory in one-against-many strategy games [9]. Finally, in [10], an evolutionary algorithm able to build FSMs is used to adaptively model opponents as the game develops, eventually using the best model to optimize its moves. While effective, this approach performs quite poorly against adversaries that cannot be modeled by FSMs.

In this paper, a different approach to evolve players for the iterated prisoner's dilemma is presented: an evolutionary algorithm is used to model and predict the opponent's behavior, while a simple brute-force algorithm select the best counterplay. The evolutionary framework encodes strategies as non-

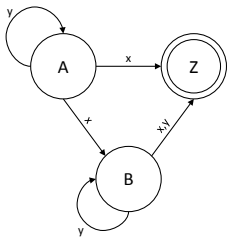
[1] Research ANd Development Corporation, a nonprofit global policy think tank formed to offer research and analysis to the United States armed forces

[2] *University of Southampton team wins Prisoner's Dilemma competition* http://www.southampton.ac.uk/mediacentre/news/2004/oct/04_151.shtml

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

deterministic finite automata, and evaluates them first on established strategies in literature, and then on their predictive capability with regards to the opponent's history of moves, building an internal model of the adversary's behavior that is at the same time *competitive*, *coherent*, and *compact*. Experimental results show that the proposed methodology is able to outperform deterministic and non-deterministic strategies, eventually using the best models to predict and counter opponent's moves.

The rest of the paper is organized as follows: section II provides the necessary concepts to introduce the scope of the work. Section III details the proposed approach, while section IV presents the experimental evaluation. Finally, section V summarizes the conclusions of the work.



	x	y
A	{B,Z}	{A}
B	{Z}	{B,Z}
Z	Ø	Ø

Fig. 1. Sample non-deterministic finite automaton $N = (\{A, B, Z\}, \{x, y\}, \Delta, A, \{Z\})$. The transition relation Δ is defined by the state transition table reported in the figure. Since both $\Delta(A, x)$ and $\Delta(B, y)$ have more than one state, N is non-deterministic.

II. Background

A. Formal definitions

The *prisoner's dilemma* is a nonzero-sum game for two players. The term *nonzero-sum* indicates that whatever benefit accrues to one player does not necessarily imply a similar penalty imposed on the other player. The prisoner's dilemma can also be defined as *non-cooperative*, to indicate that no communication is allowed between the players, apart from game actions. Nevertheless, analysis of cooperation and emerging cooperative behaviors are the typical objects of research performed on the model.

In its base form, the game describes a situation faced by two players where there are only two possible behaviors: the first can be labeled as selfless and the other selfish. If both players choose the former action, they can earn a high reward; on the other side, if both act selfishly, they will get only a low reward. But, if one is selfish and the other selfless, the first one obtains a very high reward and the second endures a penalty (very low reward, or no reward at all).

Let R be the payoff for a mutual selfless behavior (*reward*), and P the payoff if both act selfishly (*punishment*). When only one player acts selfishly, its payoff is denoted with T (*temptation*), and the payoff of its opponent with S (*sucker*). In order to make a stable cooperation preferable to an alternate series of selfish and selfless moves, *reward* is greater than the average of *temptation* and *sucker*. Thus, the requirement of the puzzle can be formalized as:

$$T > R > P > S \tag{1}$$

$$2 \cdot R > S + T \tag{2}$$

Since in the Tucker's formulation the final goal is minimizing a prison sentence instead of maximizing a reward, equations (1) and (2) are reversed in some textbooks, but the rationale is the same.

B. Non-deterministic finite automata

The first attempts to evolve strategies, back to the early 1980s, encoded individuals as simple bit vectors [11]. Bit strings have also been used for neural networks [8]. More recent solutions include lookup tables and functions generated by Genetic Programming [12]. However, finite state machines (FSMs) quickly became one of the most common ways to represent strategies [13], since they can behave like complex Markov models.

More precisely, most approaches employ *Moore* machines, that is, FSMs whose output value are determined only by the current state and not by the value of the inputs. A few works utilize FSMs where the output is a function of the current state and of the input, called *Mealy* machines. Indeed, Moore and Mealy machines are equivalent: it is trivial to translate the former in the latter, and it is always possible to translate the latter in the former by adding new states.

Indeed, while it is easy to define a player for the IPD using an FSM, it must be noted that doing so it is not always possible. Non-deterministic strategies that include random behaviors cannot be represented as FSMs, because the output is not a function of the input. Moreover, deterministic strategies that include counters or variables could require too many states to be modeled with FSMs in practice.

Many recent developments fall in these categories. For instance, Press and Dyson [14] showed the unexpected existence of *ultimatum strategies*, called "zero-determinant" (ZD). Such strategies were later proved by Adami and Hintze to be evolutionary unstable [15]; and recently, in [16], Stewart and Plotkin demonstrate that a subset of the latter strategies, called "generous", is stable in large populations. All these strategies contain *random* moves and cannot therefore be modeled using a FSM.

This paper resorts to an alternative representation, called *non-deterministic finite automaton* (NFA), or *non-deterministic finite state machine*. In automata theory, an NFA is a FSM that (i) does not necessarily require input symbols for state transitions and/or (ii) is capable of transitioning to zero or two or more states for a given start state and input symbol [17]. In a traditional FSM, all transitions are uniquely determined, and an input symbol is required for all state transitions. Although NFAs and FSMs have distinct definitions, all NFAs can be translated to equivalent DFAs using the subset construction algorithm [18], i.e., constructed DFAs and their corresponding NFAs recognize the same formal language. An example of a NFA is reported in Figure 1.

Formally, a NFA N is represented by a 5-tuple $(Q, \Sigma, \Delta, q_0, F)$, where

- Q is a finite set of states;

- Σ is a finite set of input symbols;
- Δ is a transition relation so that $\Delta : Q \times \Sigma \rightarrow P(Q)$;
- $q_0 \in Q$ is the *initial* (or *start*) state;
- $F \subseteq Q$ is a set of of *accepting* (or *final*) states.

$P(Q)$ denotes the power set of Q, that is, the set of all subsets in Q, including the empty set and Q itself.

C. Established strategies

A considerable number of different strategies for the IPD have been developed, intended as serious competitors or merely as corner cases that a candidate player should be able to deal with efficiently. What follows is a partial list of strategies that are of relevance for the present work:

TFT *(Tit for Tat)* maybe one of the most famous strategies, starts by cooperating and then copies the last move of the opponent.

CN *(Crazy Nut)* plays completely random.

TFT+ *(Super Tit for Tat)* after the first move that is completely random, replicates the opponent's previous action.

TFT- *(Unreliable Tit for Tat)* replicates the opponent's previous action with probability $p = 0.9$, otherwise plays the opposite move.

2TFT *(Two Tits For Tat)* cooperates unless the opponent defects. To retort, it defects twice. Then, if the opponent cooperated, it starts cooperating again.

TF2T *(Tit For Two Tats)* cooperates until the opponent chooses two consecutive defections. At this point, it defects and keeps on defecting until the opponent cooperates two times in a row. Then it starts cooperating again, and so on.

5TM *(Five is Too Much)*, similarly to Tit for tat, if the opponent defects the player takes a revenge in the next turn. However, when the opponent cooperates five times in a row, it defects twice.

EXT2 *(Extort2)* is a "Zero-Determinant" strategy: it imposes a linear relationship *A-P = 2(B-P)*, where *A* and *B* are respectively the payoff of this player and of his opponent, between the scores of the two player. This formula guarantees to Extort2 twice the share of payoffs above *P*, compared to those received by the opponent.

FS *(Fair Strategy)* defects and cooperates randomly, using a probability equal to the frequency of defection and cooperation moves played by the opponent.

FRT3 *(Fortress3)* is a "Group" strategy. It tries to recognize the opponent by playing the sequence DDC. If the opponent plays the same sequence, it cooperates until the opponent defects. Otherwise, if it doesn't recognize the opponent as a friend, it defects until the opponent defects on continuous two moves, then cooperates on the next.

FRT4 *(Fortress4)* is similar to Fortress3, but the handshake is a DDDC sequence and, if the opponent is not recognized, it defects until the opponent defects for three moves, then cooperates on the next.

P *(Pavlov)* starts with a cooperative move. Then, the player repeats the last action if it was profitable, i.e., if it brought an advantage over the opponent. Otherwise, it changes actions.

RL4 *(PseudoRL[4])* tries to estimate the payoff for sequences of 4 moves using an algorithm similar to reinforcement learning, then plays randomly using the values as a probability.

ZD's *(Zero Determinants)* base their next choice on its payoff in the last round. The choice to cooperate or defect is made with a certain probability for each of the four possible outcomes; in particular, we will use *ZDGTFT2 extortion* and *ZDGTFT2 fixed*, see [16] for details.

III. Proposed approach

Building on the approach published in [10], a novel evolutionary player named *Turan* is proposed. It internally encodes candidate models as Moore NFAs, and tests their behavior first by having them compete with a small set of *sparring mates*; then by comparing the output produced by the model against the previous moves of its current adversary; and finally, a parsimony metric is used to favor NFAs with as few isolated states as possible. The use of NFAs enable to overcome the main problems of [10], that is its inability to cope with strategies that cannot be modeled as FSM.

At each iteration, the best individual in the population is used as a model of the opponent's behavior in order to plan ahead, predicting its next N moves and computing the set of counter-moves that gives the best overall payoff. This parameter is listed as *Planning ahead* in Table I. Ultimately, the first move of the computed sequence is sent to the opponent. A scheme summarizing the work flow of Turan is presented in Figure 2.

A. Individual description

Individuals in Turan are Moore NFAs that all use the same set of input symbols $\{c, d\}$ (cooperate, defect), where each symbol represents a move performed by the opponent in a previous game. The set of final states F can be empty, depending on the configuration of each NFA, and is not set *a priori*. Thus, each individual i is fully described by a set of states Q_i, each one associated with an output $O_q \in \{c, d\}$, an initial state q_0 and a transition relation Δ_i, internally represented as a list of possible transitions. When two or more transitions $\Delta_1, ..., \Delta_n$ lead to different states, starting from the same state q with the same symbol σ, for simplicity the probability of each transition is set to the same value, $P(\Delta_1) = P(\Delta_2) = ... = P(\Delta_n)$.

B. Initial population

The initial population is created randomly, following a procedure that always produces valid individuals, that is, Moore NFAs with two outgoing arcs for each state, one for input *c* (cooperate), one for input *d* (defect). First, the number of states is set, following a stochastic distribution with average a and standard deviation s; then, one deterministic transitions for each input is added to each state. Finally, if every state of the NFA is reachable, the procedure ends; otherwise, all arcs

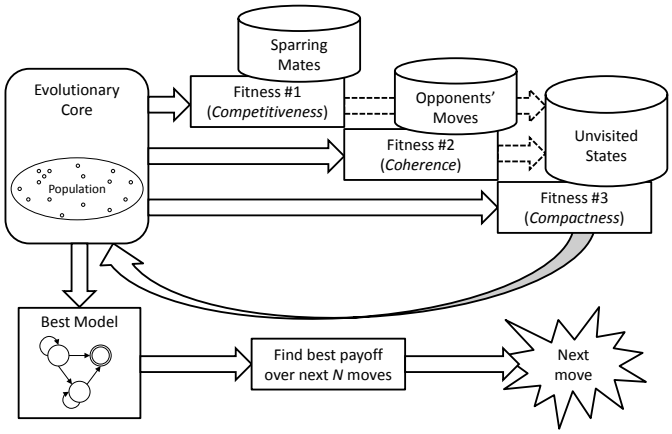


Fig. 2. Scheme of the proposed approach. The first fitness function is computed by having each individual compete against a small set of established strategies taken from literature. The second fitness function evaluates the ability of each candidate model to predict the behavior of the current opponent. The third fitness function exploits information gathered during the computation of the first two to penalize NFAs with unvisited states. At the end of each population evaluation, the best individual is used to predict the next N moves of the opponent and a corresponding sequence of N moves is generated to maximize Turan's payoff. Finally, the first move of the sequence is sent to the opponent and the loops restarts.

are deleted and replaced until a completely connected NFA is produced. As a result of this procedure, all transitions in the initial population are deterministic.

C. Individual reproduction

Individuals are selected for reproduction through a standard binary tournament selection [19]. Turan employs several kinds of mutations and a single recombination operator. Mutations' probability of activation are self-adapted over the course of a run: the probability of activating a specific mutation is proportional to the number of individuals in the current population sired by the very same operator. To avoid completely excluding some operators, every time a mutation is to be applied, a random number R in $(0, 1)$ is generated: self-adapted probabilities are used if $R \leq 0.75$; otherwise, mutations are chosen with a flat probability over all available operators.

More in particular, the mutations can:

a) Add a state: A new state with a random output (cooperate, defect) is added to the NFA, and linked to an existing state by modifying a random existing transition. Two transitions, one for cooperate and one for defect, are then added to the new state, and linked to two randomly selected states.

b) Remove a state: A randomly selected state is removed from the NFA, along with all transitions starting from it and going into it. The individual is the patched to ensure correctness.

c) Add a non-deterministic transition: A transition with input symbol σ is added to a state that already features a transition with the same input. If the former transition was the only one with symbol σ, a non-deterministic transition is created; otherwise, the previously existing non-deterministic transition can now bring the NFA to one more different state.

d) Remove a non-deterministic transition: A non-deterministic transition with input symbol σ is removed from a state.

e) Change a transition: The ending state of a transition is randomly changed to another.

f) Change an initial state: The initial state of the NFA is randomly changed to another.

g) Add an initial state: Another initial state is added to the NFA. Management of multiple initial states is described in Subsection III-D.

h) Remove an initial state: An initial state of the NFA is removed.

i) Change a state's output: The output of a randomly selected state is changed to the opposite ($\{c\}$ becomes $\{d\}$ and viceversa).

Recombination operates by adding a NFA to another, thus generating a single NFA with (at least) two initial states and two branches that are not connected. The presence of multiple initial states is taken into account during fitness evaluation, described in Subsection III-D.

D. Fitness function

Three different fitness functions are employed to achieve the objective of creating a *competitive*, *coherent* and *compact* internal model of the opponent: the first function drives the evolution towards competitive NFAs, able to deal with performing opponents taken from literature; the second rewards candidate models that most closely approximate the opponent's behavior; and the third penalizes individuals with unreachable states. The fitness functions are evaluated in lexicographical order, giving precedence to competitiveness over coherence and compactness: thus, individual A is better than individual B if its competitiveness score is better; it the competitiveness score is the same, the two individuals are then compared on coherence; and finally, if the coherence score is again the same, the two are compared on compactness.

Every time an individual is matched against a particular strategy, rules from Axelrod's second IPD tournament are used. The score for each player is the sum of the payoffs obtained over 5 games, and after each move (that is, the choice to cooperate or defect) the probability that the game will continue is set to $p_c = 0.99654$. In this way, strategies cannot exploit regularities in games made of a fixed number of moves. This choice is also coherent with recent software developed specifically for running IPD tournaments [20].

1) Competitiveness: One of the assumption of the proposed approach is that the opponent is actually competitive. Creating a model of the adversary from scratch, considering only its past moves, could lead the evolutionary algorithm to creating overfitted NFAs with one state per move. Such a model would have no predictive capability.

Since an opponent can be assumed to be quite competitive, the first fitness function has the aim of favoring the emergence of compact and competitive individuals. In particular,

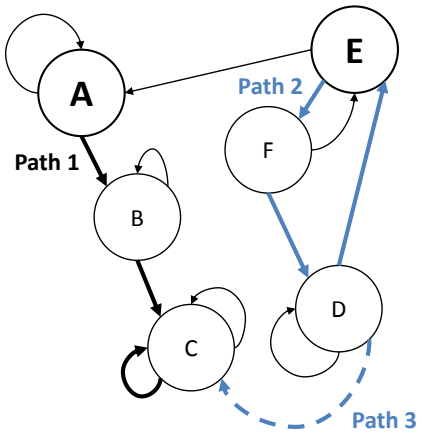


Fig. 3. Example of concurrent exploration of multiple paths in a NFA. Both A and E are initial states, giving birth to **Path 1** (arrows in bold black) and **Path 2** (arrows in bold grey). State D has a non-deterministic transition, so all its possible destination states are explored, creating a new **Path 3**, that is overlapped with Path 2 until state D.

each candidate model is matched against 5 sparring mates, established strategies taken from literature. Since individuals in the EA are non-deterministic and can feature multiple initial states, multiple paths are explored at the same time and the next move is randomly drawn from the set of outputs of all current states. See Figure 3 for an example.

The 5 sparring mates are CN, TFT+, TFT-, TF2T and 2TFT (described in II-C). Different games are played against each each sparring mate, the probability of continuing one of these games after each move is $1/10$ of the corresponding probability for standard games, that is $p = 0.099654$.

2) Coherence: Each candidate NFA is compared against the known history of moves performed by the current opponent, and rewarded accordingly, depending on the accuracy of its predictions. Since individuals are non-deterministic and can feature more than one initial state, multiple paths are explored at the same time, and multiple sequences of output are produced. The sequences are compared to the opponent's behavior, and if at least one of the sequences matches, the NFA is considered coherent. For computational efficiency, the simulation is not exact and does not take into account rejoining paths. An example is portrayed in Figure 3.

The total number of paths is considered to strongly discourage non-deterministic models. Turan introduces non determinism when strictly essential to model the opponent. Moreover, to take into account the eventuality that the opponent is also adapting its strategy, the coherence is weighted: being coherent in the recent moves of the game is more important than being coherent in the old ones.

3) Compactness: The visited states are marked during the evaluation of the previous fitness functions, and NFAs that present unvisited states are penalized, depending on the number of unreached states. Moreover, a small penalty is added considering the overall number of states.

TABLE I. PARAMETERS USED DURING THE EXPERIMENTAL EVALUATION.

Parameter	Value
Evolutionary algorithm	
Population size	50
Offspring size	20
Replacement strategy	$\mu + \lambda$
Initial population	
Initial population size	100
Initial states (average)	5
Initial states (non determinism)	0.2
Payoff	
P (punishment)	1
R (reward)	3
S (sucker)	0
T (temptation)	5
Games	
Planning ahead	10 moves
Number of games	3
Probability of continuing a game after each move	0.99654

E. Extinction

From preliminary runs, it becomes evident how attaining *coherence* is the hardest task for the evolutionary process. The population might be invaded by high-performing NFAs with almost null representativeness of the current opponent. In order to avoid this problem, if at the beginning of a generation no individual has a coherence fitness value higher than $\epsilon_{coherence}$ and there has been $n_{stagnation}$ generations with no improvement of the best individual, an *extinction* procedure is performed, taking inspiration from [21]. In particular, only λ individuals are preserved, while the rest of the population is filled with randomly-generated NFAs.

IV. EXPERIMENTAL RESULTS

Turan is first matched against the well-known *Tit for Tat*, with the objective to verify its opponent modeling capabilities. Subsequently, to assess its efficiency, it is played against several established strategies. During all experiments, the evolutionary framework is configured with parameters reported in Table I.

A. Modeling the TFT

In the first part of the experiments, Turan is matched against the TFT. A single run of Turan is executed, spanning 3 different games, and the best individuals is visually inspected at regular intervals. Figure 4 shows the progression, with four individuals selected at pivotal points over the generations.

It is interesting to notice how Turan starts with an extremely complex, non-deterministic NFA that nevertheless captures the TFT predisposition to cooperation: individual `a`'s state 1 has output *c* and a loop that returns in the state if the opponent cooperates. In a second phase, Turan oversimplifies TFT's behavior, with a minimal NFA (`b`) assimilable to an AC. In a third phase the EA extendes its internal representation again, trying to model the outcome of defections, while maintaining the good building block of continual cooperation (state 1 in `c`). Finally, in the last part of the evolution, Turan stably converges on the correct model (`d`).

B. Established strategies

In this second tranche of experiments, Turan is tested against effective strategies, both deterministic and non-

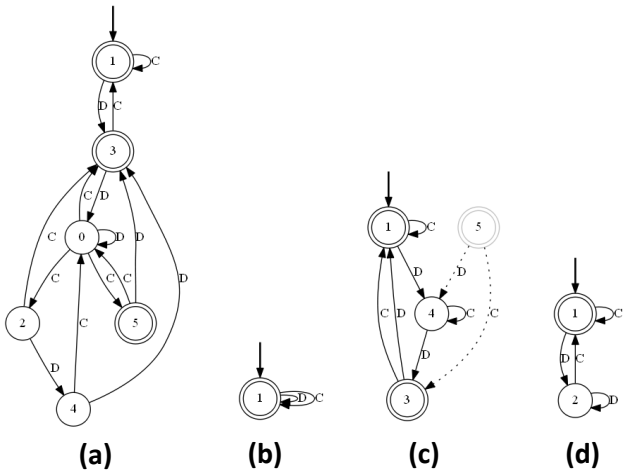


Fig. 4. Opponent's model created by Turan over an evolutionary run against TFT, from the first generations (left) to the end of the evolution (right). Here, states with a single double circle have output *cooperate*, while states with a single circle have output *defect*.

deterministic. It is interesting to notice that, while some of them can be precisely modeled as NFAs, the behavior of others can only be approximated to a NFA. A single run of Turan is executed for each opponent, spanning 3 different games. Results are summarized in Table II: since the number of moves in each game may vary, the average points-per-move is reported.

Against relatively simple strategies, such as Pavlov, 5TM and CN, Turan is able to completely learn the opponent's behavior during Game 1, subsequently maximizing its behavior. Even with the TFT+, the EA is able to find the best solution and steadily increase its payoff. Interestingly, Turan is also able to quickly detect the handshake sequences of both FTR3 and FTR4, exploiting their internal models to quickly gain an edge.

When dealing with simple adaptive strategies, such as PseudoRL[4], the proposed approach is able to outperform the adversaries from the beginning. The Fair Strategy proves to be a more difficult opponent: after a close loss in Game 1, Turan manages to win by a short margin in Game 2, only to lose again in Game 3. Results against this strategy need further analysis, in order to better understand the interactions between the two adaptive opponents. ZDs and EXT2 are by far the hardest matches: against extortion strategies, Turan is able to slowly increase its total payoff, but three games are not enough to overcome the opponents, albeit the proposed approach comes close to the objective against EXT2.

V. Conclusions

This paper presents Turan, an evolutionary player for the Iterated Prisoner's Dilemma that is able to build an internal model of its opponent as the game goes on, eventually exploiting the model to predict the adversary and maximize its payoff. Since internal models are Non-deterministic Finite Automata, Turan is able to model non-deterministic strategies as well as deterministic ones.

Preliminary experimental results show that the proposed approach is able to correctly model a *Tit-for-tat* opponent over the course of three games, and it performs well against several established strategies, losing only against extortion and zero-determinant adversaries.

A thorough analysis of new results and their implications is currently under way, as well as the run of complete tournaments using Oyun [20], where Turan can measure its overall performance against other effective strategies.

References

[1] E. N. Zalta and S. Abramsky, "Stanford encyclopedia of philosophy," 2003.

[2] W. Poundstone and N. Metropolis, "Prisoner's dilemma: John von neumann, game theory, and the puzzle of the bomb," *Physics Today*, vol. 45, p. 73, 1992.

[3] R. J. Aumann, "Acceptable points in general cooperative n-person games," *Contributions to the Theory of Games*, vol. 4, pp. 287–324, 1959.

[4] R. Axelrod, "Effective choice in the prisoner's dilemma," *Journal of Conflict Resolution*, vol. 24, no. 1, pp. 3–25, 1980.

[5] R. M. Axelrod, *The complexity of cooperation: Agent-based models of competition and collaboration*. Princeton University Press, 1997.

[6] G. Kendall, X. Yao, and S. Y. Chong, *The Iterated Prisoners' Dilemma: 20 Years on*. World Scientific Publishing Co., Inc., 2007.

[7] D. B. Fogel, "Evolving behaviors in the iterated prisoner's dilemma," *Evolutionary Computation*, vol. 1, no. 1, pp. 77–97, 1993.

[8] N. Franken and A. Engelbrecht, "Particle swarm optimization approaches to coevolve strategies for the iterated prisoner's dilemma," *Evolutionary Computation, IEEE Transactions on*, vol. 9, no. 6, pp. 562–579, 2005.

[9] A. Ghoneim, G. Greenwood, and H. Abbass, "Distributing cognitive resources in one-against-many strategy games," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, June 2013, pp. 1387–1394.

[10] A. Uthor, "An article," in *A Conference*, 2011.

[11] R. Axelrod and W. D. Hamilton, "The evolution of cooperation," *Science*, vol. 211, no. 4489, pp. 1390–1396, 1981.

[12] J. R. Koza, *Genetic Programming: vol. 1, On the programming of computers by means of natural selection*. MIT press, 1992, vol. 1.

[13] A. Rubinstein, "Finite automata play the repeated prisoner's dilemma," *Journal of economic theory*, vol. 39, no. 1, pp. 83–96, 1986.

[14] W. H. Press and F. J. Dyson, "Iterated prisoner's dilemma contains strategies that dominate any evolutionary opponent," *Proceedings of the National Academy of Sciences*, vol. 109, no. 26, pp. 10 409–10 413, 2012. [Online]. Available: http://www.pnas.org/content/109/26/10409.abstract

[15] C. Adami and A. Hintze, "Evolutionary instability of zero-determinant strategies demonstrates that winning is not everything," *Nature Communications*, vol. 4, p. Online, August 2013.

[16] A. J. Stewart and J. B. Plotkin, "From extortion to generosity, evolution in the iterated prisoner's dilemma," *Proceedings of the National Academy of Sciences*, 2013. [Online]. Available: http://www.pnas.org/content/early/2013/08/28/1306246110.abstract

[17] M. O. Rabin and D. Scott, "Finite automata and their decision problems," *IBM journal of research and development*, vol. 3, no. 2, pp. 114–125, 1959.

[18] J. C. Martin, *Introduction to Languages and the Theory of Computation*. McGraw-Hill New York, 2003, vol. 2.

[19] A. Brindle, "Genetic algorithms for function optimization," Ph.D. dissertation, Edmonton: University of Alberta, Department of Computer Science, 1981.

[20] C. H. Pence and L. Buchak, "Oyun: A new, free program for iterated prisoner's dilemma tournaments in the classroom," *Evolution: Education and Outreach*, vol. 5, no. 3, pp. 467–476, 2012.

TABLE II. RESULTS FOR TURAN AGAINST ESTABLISHED STRATEGIES. SINCE EACH GAME FEATURES A DIFFERENT NUMBER OF MOVES, THE AVERAGE POINTS PER MOVE ARE REPORTED. THE COMPLEXITY IS THE NUMBER OF DIFFERENT STATES IN THE BEST NFA AT THE END OF THE EVOLUTIONARY PROCESS.

Strategy	Game 1		Game 2		Game 3		Complexity
	Turan	Opponent	Turan	Opponent	Turan	Opponent	
5TM (*Five is too much*)	2.738	2.023	2.786	1.696	2.781	1.984	42
CN (*Crazy Nut*)	2.530	1.370	2.886	0.936	2.467	1.569	16
EXT2 (*Extort2*)	0.994	1.024	1.008	1.217	1.108	1.222	7
FS (*Fair Strategy*)	1.911	2.208	2.869	2.855	1.085	1.984	17
FRT3 (*Fortress3*)	2.946	2.619	2.950	2.825	2.765	2.307	46
FRT4 (*Fortress4*)	2.185	0.815	2.259	0.755	2.281	0.729	10
P (*Pavlov*)	2.333	1.917	2.504	1.655	2.608	1.644	18
RL4 (*PseudoRL[4]*)	4.738	0.095	4.933	0.017	4.739	0.065	7
TFT+ (*Super Tit for Tat*)	2.881	2.881	2.866	2.866	2.971	2.971	11
ZDGTFT2 extortion	0.994	1.024	1.047	1.228	1.072	1.578	9
ZDGTFT2 fixed	2.036	2.690	2.019	2.841	1.941	2.497	33

[21] G. Greewood, G. B. Fogel, and M. Ciobanu, "Emphasizing extinction in evolutionary programming," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 1. IEEE, 1999.