Complex Systems **2** (1988) 663–670

Simulating the Evolution of Behavior: the Iterated Prisoners' Dilemma Problem

David M. Chess*
Computing Systems Department, IBM T. J. Watson Research Center, Post Office Box 218, Yorktown Heights, NY, USA

Abstract. A system is described in which a number of artificial agents (represented by simple mathematical expressions) compete for the right to "reproduce" (that is, to cause new agents with similar properties to be generated). By simulating some of the essential features of biological evolution, the system makes possible some novel insights into the behavior of communities of agents over time. The results of Fujiki and Dickinson on the Iterated Prisoners' Dilemma problem (IPD) are essentially confirmed. The typical course of evolution of a community of IPD players is described, and possibilities for further work are outlined. This study is also relevant to machine learning, and adaptive systems in general.

1. Introduction

The theory of evolution provides a broad, powerful model for a process by which complex and capable entities (including human beings) can arise as the result of uncaring non-sentient forces.[1] The kinds of computations typical of current digital computers are paradigmatically non-sentient. For this reason (and no doubt for others) there has been a continuing undercurrent of interest in the possibility of arriving at complex (perhaps even intelligent) systems by implementing evolutionary models on digital computers.

Fogel, Owens, and Walsh [3] present a system in which entities modelled as finite automata evolve to satisfy certain capability criteria, and speculate on the implications of such systems for natural and artificial intelligence. John Holland and his associates [4,5] have developed a powerful mathematical theory of certain types of evolving systems, suitable for computer implementation. Dress [6] ties together the ideas of evolution and neural networks, by presenting a system in which networks of simulated neurons evolve to satisfy criteria involving learning ability. Brown [7] incorporates evolutionary

*Network addresses: `chess@ibm.com` (internet), `CHESS at YKTVMV` (Bitnet).

[1] Richard Dawkins [1,2] is one of the primary modern expositors of this idea.

	A cooperates	A defects
B cooperates	$p(A) = 1, p(B) = 1$	$p(A) = 2, p(B) = -1$
B defects	$p(A) = -1, p(B) = 2$	$p(A) = 0, p(B) = 0$

Table 1: $p(A)$ and $p(B)$ are payoffs to two players, A and B, in the Prisoner's Dilemma.

ideas into the construction of rules for cellular automata. On the popular front, A. K. Dewdney [8] has recently tickled the general reader with ideas about simulating the evolution of form (the length of a dinosaur's neck, and of the spines on a spiny plant) on small personal computers.

This paper presents some principles of and results from a system which simulates the evolution of behavior in simple tasks (tasks in which the entities have input systems that pass only small amounts of information, and output systems that limit them to a small number of possible actions). Despite the simplicity of the tasks studied, the events that occur as the entities evolve can be quite complex and interesting. The particular situation faced by the entities described in this paper is a version of the Prisoner's Dilemma problem.

2. The prisoner's dilemma

The Prisoner's Dilemma is a classic example in game theory (attributed, probably correctly, to A. W. Tucker; see for instance [9]). A typical Prisoner's Dilemma game has a payoff matrix as in table 1.

The two strategies are commonly called **cooperate** and **defect**. The game is completely symmetric; each player receives a payoff of 1 if he and his partner both cooperate, 0 if both defect, -1 if he cooperates and his partner defects, and 2 if he defects and his partner cooperates. In game-theoretic terms, defecting completely dominates cooperation; that is, no matter what one's partner does, one's own payoff is higher if one defects. On the other hand, the highest *total* payoff is obtained when both parties cooperate, and the payoff to each party in that case is in fact higher than if both parties behave "rationally" and defect.

A particularly interesting case of this situation is the "Iterated" Prisoner's Dilemma (**IPD**). In this form of the problem, the parties play the game a number of times, and each party can decide how to behave according to how the other party has behaved in previous encounters. Strategies for the iterated case have been studied extensively by Axelrod [10]. In one study, he solicited iterated-prisoner's-dilemma algorithms from many sources, and ran them in a competitive environment, in which the highest-scoring algorithms were allowed to "reproduce," and the lowest-scoring ones were eliminated.[2]

The algorithms that do best at the iterated prisoner's dilemma are those which (roughly) will cooperate with others like themselves, but will not co-

[2]This sort of competition is similar to simulated evolution, except that no mutations occur; that is, no algorithms arise except those explicitly given by the humans involved.

operate with others which do not cooperate. Axelrod summarizes the characteristics of successful algorithms; three of the most important are that they are

Nice Will begin by cooperating, and not defect if the partner is also Nice.

Provocable Will not continue to cooperate with a partner that defects too often.

Forgiving Will eventually return to cooperation if a formerly-defecting partner begins to cooperate.

The simplest successful algorithm is "Tit-for-Tat" (TFT), which begins by cooperating, and continues by doing each turn what its partner did last turn. It has all three of Axelrod's desirable qualities, and does very well at the task. A community of TFT players will always cooperate with one another, but since each one will retaliate immediately in response to defection, a defector cannot prosper by exploiting any of them.

3. The IPD and evolution

The iterated prisoner's dilemma is a simple game, with few inputs (essentially only the memory of prior rounds), and a single binary output (Cooperate or Defect). Algorithms to play the game are thus simple to represent, and lend themselves well to efficient implementation in a digital computer. On the other hand, the game has interesting and non-trivial properties, as well as (at least metaphorical) relevance to a variety of human problems. Entities represented by IPD algorithms are therefore very good candidates for the computer simulation of evolution.

Fujiki and Dickinson [11] present an evolutionary system in which a subset of LISP S-expressions are used to represent IPD players. In their experiments, the algorithms that evolved included a variant of Tit-for-Tat, which would defect only if its partner defected twice consecutively,[3] and a *nice-but-unforgiving* algorithm, which would cooperate as long as its partner never defected, but which would always defect if its partner ever had.

4. The system

The system described in the present paper uses expressions in integer arithmetic to represent IPD algorithms. Each entity is an arithmetic expression in terms of three variables:

[3]In fact, it would also defect on the last round of a game. It is interesting to note that a population of Tit-for-Tat players can be successfully invaded by a new "species" which plays TFT until the last round, and then defects. This population can in turn be invaded by a species which plays TFT until the second-to-last round, and then defects, and so on. At some point on this "slippery slope" of invasions, there will be a population which can be invaded by a pair of pure TFT players; in this sense, fitness (or "invadability") is intransitive in the IPD; every strategy in the set containing TFT and its "defect at the end" variants can be invaded by at least one of the others.

LastTime: What action the partner took last round (set to 1 if the action was Cooperate, −1 otherwise)

DefectCount: How many times in the past the partner has defected

CoopCount: How many times in the past the partner has cooperated

Each variable may appear more than once, or not at all, and the expressions are not necessarily linear in any of them. To determine an entity's behavior for the current round, the expression representing the entity is evaluated for the current values of the variables, and the entity's behavior is taken to be Defect if the value of the expression is less than 0, and Cooperate otherwise.

The simplest expressions corresponding to some familiar IPD players are

Tit-for-Tat: $(LastTime)$

Nice but Unforgiving: $(0 - DefectCount)$

Always Cooperate: (1)

Always Defect: (−1)

although much more complex expressions are possible; for instance,

$$(37 * \text{LastTime} + (\text{DefectCount} - \text{DefectCount}) - 6 * \text{LastTime} * \text{LastTime})$$

is behaviorally equivalent to simply (LastTime), and is therefore a representation of Tit-for-Tat. Other more complex expressions represent more complex behaviors; a randomly generated expression is likely to represent more or less irrational behavior.

The current system uses a fixed number of entities, which are initialized to random expressions of the variables (the expressions are stored internally as parse trees). In each generation the entities are grouped into pairs, and each pair plays a few unscored "startup" rounds of the game, followed by a number of scored rounds. When all pairs have played, the lowest-scoring entities are replaced by copies of the highest-scoring ones. To introduce variation, the copying process is imperfect. During a copy, there is a small chance that some subtree of the source expression will be replaced by a randomly-generated new subtree when copied to the destination. Unlike Fujiki and Dickinson's system, this system has nothing corresponding to genetic "crossing-over" or "inversion."

5. Results

A typical run of the system displays four distinctive stages. In the first, which might be called the "Era of Exploitation," habitual defectors dominate. This is because, in the typical random mix with which the system starts, most entities are either pure defectors, pure cooperators, or erratic players whose behavior is effectively random. In a population with this mix, the defectors do very well, at the expense of the cooperators especially, but also at the expense of the erratic players.

The second stage appears when virtually all of the "exploitable" players (the pure cooperators and the erratics) have been eliminated. In this stage, most of the interactions are between pairs of pure defectors, and total scores tend to be low. This stage might be called the "Nadir."

The third stage begins as rational entities (those with some of Axelrod's desirable qualities) begin to multiply. If some of these were present at the beginning, and survived the Era of Exploitation, the third stage will begin almost as soon as the Nadir is reached. If not, the Nadir will continue until two or more arise by mutation. In any case, during this stage, which might be called the "Growth of Trust," exploiters begin to be eliminated by rational entities. The growth of rational entities tends to be geometric, since every time two rational entities meet, they will cooperate, scoring higher than any pure defector can, and will therefore produce more rational offspring for the next generation. In the next generation, the probability of two rational entities meeting will be that much higher, and the effect will snowball.

Eventually, all the pure defectors from the Nadir stage have been eliminated, and (except for "sports" which appear by mutation) all the entities in the system are rational. At this stage, which might be called "Equilibrium," cooperation is the rule, and total scores are high. Entities observed in this stage are primarily Tit-for-Tat players and Nice-but-Unforgiving players. Some examples of Tit-for-Tat players present in the equilibrium stages of typical runs of the system are

```
LastTime

(LastTime + LastTime)

(DefectCount mod 7) * LastTime
```

Some Nice-but-Unforgiving players are

```
(-28 * DefectCount)

((CoopCount - 13) - CoopCount) * DefectCount

(LastTime - DefectCount)
```

When a defector arises by mutation in the equilibrium stage, it is quickly eliminated, since as soon as it is paired with a rational entity, that pair will score only the defect-defect payoff, while the other entities are scoring the cooperate-cooperate payoff. When a cooperator arises by mutation, it can

survive for a longer time, but eventually it will be paired with a newly-arisen defector, and eliminated.

Studies with this system therefore essentially confirm the work of Fujiki and Dickinson, and of Axelrod. In both the Fujiki and Dickinson work and the present work, the Forgiving characteristic of Axelrod did not play a large role. Unforgiving algorithms did as well as forgiving ones; algorithms in the equilibrium states were Nice and Provocable, but many were not Forgiving. This is almost certainly because of the simplicity of the systems, and the comparatively short runs. The Forgiving characteristic is most useful when a significant number of other entities in the system employ complex "feeling-out" strategies and will revert to cooperation when an initial defection provokes retaliation. Such complex behaviors have not been observed in the present system, so the Forgiving characteristic has probably not been selected for.

6. Notes and further work

The basic results of this work, including the stages of evolution and the behaviors observed in equilibrium, are quite robust. That is, they are not essentially changed by varying various parameters of the system. Essentially identical results were obtained with various settings of parameters such as mutation rate, number of entities in the system, details of the payoff matrix, number of rounds of iteration in the IPD, and so on.[4] The same results were also achieved when the entities were constrained to be expressions linear in each of the three variables. These facts, and the fact that the independent work of Fujiki and Dickinson also produced very similar results, suggests that the observed phenomena are truly essential to the IPD itself, and not simply artifacts of the implementation.

The system as it exists today can be used to study tasks other than the IPD. Studies are currently in progress on other simple game-theoretic tasks, such as coordination and anti-coordination problems, and variations on the Prisoner's Dilemma. The system has also been extended to include more complex entities, faced with more complex environments, and capable of more complex behaviors. One study involves entities which perceive an environment containing both "food" and other entities, and which have a variety of behaviors to choose from, including moving towards or away from any object in the environment, attempting to "eat" any object in the environment, and so on.

All the behaviors so far studied are fixed at the entity level; that is, an entity will always behave the same way when given the same input, and no learning occurs in an individual entity (although the system as a whole

[4]Most of the studies were done in a system with 100 entities, 5 non-scored rounds, 50 scored rounds, a mutation probability of 0.3% at each node in a parse tree, and the replacement of the 10 lowest-scoring entities by copies of the 10 highest-scoring ones at the end of each generation. The system used for most of the work was an IBM PC/AT, running software written in IBM PC Pascal.

may be said to be learning). One obvious extension to the system would be to implement entities which store the results of previous behaviors and use some sort of pattern-matching algorithm to choose a new behavior based on the stored memories. Evolution would then occur on the higher-level parameters governing the behavior of the memory and of the pattern-matching algorithms. It is an open question whether the sort of evolution described above can produce interesting examples of more complex behaviors in feasible amounts of run time.

7. Conclusions

The system described here confirms some previous work on the evolution of Iterated Prisoners' Dilemma players, and describes the stages of evolution through which the players typically pass. Because of the obvious biological, and even human, analogies that they suggest, systems like this can be exciting to work with, and it is important to avoid exaggeration. Evolving systems are not necessarily intelligent systems; the time required for any current system to evolve a database program, let alone a natural-language-understanding program, would be prohibitive.[5] On the other hand, these systems represent an approach to the production of behavior which is both different from traditional program-development approaches, and similar to the processes that produced (for example) all of us. Continuing research into them cannot fail to produce worthwhile insights, both into the theory of the tasks studied (such as the IPD), and into the mechanics of the evolutionary process itself.

References

[1] Richard Dawkins, *The Selfish Gene*, (Oxford University Press, New York, 1976).

[2] Richard Dawkins, *The Blind Watchmaker*, (Norton, New York, 1986).

[3] Lawrence Fogel, Alvin Owens, and Michael Walsh, *Artificial Intelligence through Simulated Evolution*, (John Wiley and Sons, New York, 1966).

[4] John H. Holland, *Adaptation in Natural and Artificial Systems*, (University of Michigan Press, Ann Arbor, 1975).

[5] John H. Holland, "Genetic Algorithms and Classifier Systems: Foundations and Future Directions" in *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, ed. John J. Grefenstette, (Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey) 82–89.

[6] W. B. Dress, "Darwinian Optimization of Synthetic Neural Systems," *ICNN San Diego IEEE*, (June 21, 1987) 1–7.

[5] See [12] for some thoughts on the relationship of evolution to artificial intelligence.

[7] David B. Brown, "Competition of Cellular Automata Rules," *Complex Systems*, **1** (1987) 169–180.

[8] A. K. Dewdney, "Computer Recreations," *Scientific American*, (February, 1988) 128–131.

[9] Morton D. Davis, *Game Theory*, (Basic Books, New York, 1970).

[10] Robert Axelrod, *The Evolution of Cooperation*, (Basic Books, New York, 1984).

[11] Cory Fujiki and John Dickinson, "Using the Genetic Algorithm to Generate LISP Source Code to Solve the Prisoner's Dilemma" in *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, John J. Grefenstette, ed., (Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey) 236–240.

[12] A. J. Fenanzo, Jr., "Darwinian Evolution as a Paradigm for AI Research," *ACM SIGART Newsletter*, **97**, (July 1986) 22–23.