A Game-Theoretic Memory Mechanism for Coevolution *

Sevan G. Ficici and Jordan B. Pollack

Department of Computer Science
Brandeis University
Waltham Massachusetts 02454 USA
www.demo.cs.brandeis.edu

Abstract. One problem associated with coevolutionary algorithms is that of *forgetting*, where one or more previously acquired traits are lost only to be needed later. We introduce a new coevolutionary memory mechanism to help prevent forgetting that is built upon game-theoretic principles, specifically Nash equilibrium. This "Nash memory" mechanism has the following properties: 1) It accumulates a collection of salient traits discovered by search, and represents this collection as a *mixed strategy*. 2) This mixed strategy monotonically approaches the quality of a Nash equilibrium strategy as search progresses, thus acting as a "ratchet" mechanism. 3) The memory naturally embodies the result (solution) obtained by the coevolutionary process. 4) The memory appropriately handles intransitive cycles (subject to resource limitations). We demonstrate our Nash memory using Watson and Pollack's intransitive numbers game, and compare its performance to the conventional "Hall of Fame" memory and the more recently proposed Dominance Tournament.

1 Introduction

Among the problems one often confronts when using a coevolutionary algorithm is that of *forgetting*, where one or more previously acquired traits (i.e., components of behavior) are lost only to be needed later, and so must be re-learnt. Since selection pressure determines which traits have the opportunity to reproduce, the disappearance of a trait has but a few explanations. First, a trait is selected *against* when individuals with that trait are less fit, on average, than individuals without it. Second, a trait is subject to *drift* when individuals with that trait are equally fit, on average, as individuals without it. This drift may occur in either of two ways (or both): Due to sampling error in the population dynamics (where fewer of the individuals with the trait actually reproduce, thus reducing their numbers and increasing the risk of extinction for that trait), and due to variational biases that cause the trait to be lost in the generation of offspring. (Even if a trait is selected for, the variational process may be strongly

* E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2723, pp. 286-297, 2003.

biased against it, thus causing offspring to lack the trait; such variational bias is the impetus behind the technique of *elitism* that is often used in evolutionary algorithms [6].) The disappearance of a trait becomes an instance of forgetting if the discarded trait is subsequently able to contribute *positively* to the fitness of an individual lacking that trait.

Such variability in a trait's contribution to fitness is possible in coevolution because the observed quality of an individual (and the traits it carries) may be highly contingent upon the context of its evaluation—the population of coevolving individuals. Thus, at one moment in evolutionary time, a trait may be highly undesirable (or of neutral worth but difficult to maintain) and purged, only to become of value at some later point in time. Simple zero-sum games with intransitivities illustrate how contingency can cause a coevolutionary system to learn, forget, and re-learn traits in a cyclic fashion; the familiar Rock-Paper-Scissors game is the canonical example of an intransitive cycle (Rock beats Scissors, which beats Paper, which in turn beats Rock), where one strategy exploits the next, leading to a perpetual alternation of selective pressure for and then against each strategy in turn.

The problem of designing a heuristic "memory mechanism" to prevent forgetting intrinsically entails the problem of deciding what it is that we are trying to "remember"—what is our *solution concept*? That is, what collection of traits constitutes the desired or "correct" set, and what properties does this collection have? A principled answer to this question is imperative when a domain forces mutual exclusivity between certain traits, or when an evolutionary representation (genome) cannot simultaneously encode all desired traits. Once we have our solution concept, what organizing principle do we use in the memory mechanism to obtain it? Thus, we view a memory mechanism as an accumulator of traits; a trait enters and remains in the memory only if it is "worth" remembering, according to our organizing principle.

In this paper, we obtain our solution concept—Nash equilibrium—from game theory, and use game theory to construct an organizing principle for a new memory mechanism for coevolution. Using Watson and Pollack's *intransitive numbers game* [13], which is rife with intransitive cycles, we demonstrate the ability of our "Nash memory" mechanism to approximate a monotonic and asymptotic convergence to the Nash equilibrium of the game. Further, we show that the commonly used "best of generation" memory mechanisms (e.g., Rosin and Belew's *Hall of Fame* [9]), as well as Stanley and Miikkulainen's more recently proposed *Dominance Tournament* [11] mechanism, do not generally accumulate a collection of traits that corresponds to Nash equilibrium.

The paper is organized as follows. Section 2 reviews key concepts from game theory. Section 3 continues with additional concepts such as *security* and *domination*. Section 4 then reviews the literature on methods to detect and prevent forgetting. Section 5 details the construction and operation of our Nash memory mechanism. Section 6 defines and discusses the properties of the intransitive numbers game. Section 7 presents our experimental setup and results. Section 8 offers concluding remarks.

2 Game Theory Fundamentals

Here we present some fundamental points on game theory [5]. Due to limited space, we must restrict our discussion to symmetric zero-sum games (in strategic form) for two players; nevertheless, our Nash memory mechanism is easily generalized to asymmetric constant-sum games for two players.

Being symmetric, the game $\mathbf{G}$ defines a single set of *pure strategies*, $\mathcal{S}$, that is made available to both players. A player may adopt a pure strategy $s \in \mathcal{S}$ or may instead play a *mixed strategy*, which is specified by a probability distribution over $\mathcal{S}$. Pure strategies played with non-zero probability by a mixed strategy m are *in support* of (also known as "carriers" of) m. The function $\mathrm{C}(m)$ returns the set of pure strategies that support the mixture m; hence, $\mathrm{C}(m) = \{s \in \mathcal{S} : \Pr(s|m) > 0\}$. A pure strategy can be understood as a degenerate mixture.

For any pair of strategy choices made by the two players, the game $\mathbf{G}$ specifies the expected payoff earned by each player; by convention, $\mathrm{E}(\alpha, \beta)$ denotes the expected payoff earned by strategy α when played against β. Since the game $\mathbf{G}$ is zero-sum, $\mathrm{E}(\alpha, \beta) + \mathrm{E}(\beta, \alpha) = 0$. This implies that $\mathrm{E}(\alpha, \alpha) = 0$.

Any pure or mixed strategy s^* that is its own "best response" is a *Nash equilibrium* strategy. More precisely, s^* is a Nash iff $\forall$ mixtures $m : \mathrm{E}(s^*, s^*) \geq \mathrm{E}(m, s^*)$. That is, if one player plays s^*, then the highest payoff obtainable by the other player is received by also playing s^*. Thus, a Nash strategy guarantees that $\forall$ mixtures $m : \mathrm{E}(s^*, m) \geq 0$. Due to this property, Nash strategies provide the maximal *security level* that can be obtained in a zero-sum game—no other solution concept can guarantee a higher expected payoff without regard to the opponent's strategy choice. This security level is also known as the *value* of the game. Finally, all games with finite $\mathcal{S}$ have at least one Nash equilibrium.

3 Additional Concepts

The *security set* of a mixture m is the set of pure strategies against which m earns an expected payoff greater than or equal to zero; that is, $\mathrm{S}(m) = \{s \in \mathcal{S} : \mathrm{E}(m, s) \geq 0\}$. If the security set of a mixture contains the mixture's support set, then the mixture is *support-secure*; mixture m is support-secure iff $\mathrm{S}(m) \supseteq \mathrm{C}(m)$. All pure strategies are trivially support-secure. For any pure or mixed Nash strategy s^*, $\mathrm{S}(s^*) \supseteq \mathrm{C}(s^*)$ and $\mathrm{S}(s^*) = \mathcal{S}$. The *vulnerability set* of a mixture m is the complement (with respect to $\mathcal{S}$) of the security set: $\mathrm{V}(m) = \overline{\mathrm{S}(m)}$.

One strategy α *dominates* another strategy β iff $\forall s \in \mathcal{S} : \mathrm{E}(\alpha, s) \geq \mathrm{E}(\beta, s)$ and $\exists s \in \mathcal{S} : \mathrm{E}(\alpha, s) > \mathrm{E}(\beta, s)$. By definition, for any strategy s and any Nash strategy s^*, $\mathrm{E}(s^*, s) \geq 0$; therefore, any strategy that dominates a Nash must be Nash, as well. At the same time, Nash strategies need not dominate all (or any) non-Nash strategies. We examine this point further, below.

4 Memory Methods and Solution Concepts

Research to prevent forgetting includes a number of *memory mechanisms* that maintain a collection of "good" individuals (according to some organizing prin-

ciple) discovered over evolutionary time; the memory thus encapsulates a wider range of phenotypes than is typically found in the coevolving population at any one time. The additional phenotypic variety afforded by the memory is used to augment the evaluation process, broaden selection pressure, and thereby reduce the likelihood of forgetting. (Note that memory mechanisms are often identical to methods used to detect forgetting [2, 7, 11]. Some domains, however, permit the use of an external objective metric of behavior [4], or allow easy comparison to a known optimal solution [8], as alternative ways to detect forgetting.)

Almost all memory mechanisms in the literature are instances of a general "best of generation" (BOG) model where 1) the most fit individual in each of the m most recent generations is retained by the memory mechanism and 2) l of the m ($l \leq m$) retained individuals are sampled without replacement for use in testing individuals in the current generation. Examples of this approach are found in Sims [10] ($m = 1$, $l = 1$), Rosin and Belew [9] (e.g., $m = \infty$, $l = 20$), and Nolfi and Floreano [7] ($m = 10$, $l = 10$).

In a contrasting approach, Stanley and Miikkulainen [11] propose that their *dominance tournament* (DT)—a method intended to monitor coevolutionary progress—can be adapted for use as a memory mechanism. The principle is to retain the most fit individual of the current generation only if it beats all the individuals previously retained by the memory; that is, the "generation champion" must dominate the contents of the memory to be included. (Note, however, that domination is determined only with respect to the memory's contents.) Thus, no intransitive cycles can exist amongst the strategies in the memory.

As we state above, deciding how to organize a memory entails deciding what kind of solution we wish to obtain. If we view the memory as a repository of the most salient traits discovered so far, then the logical extension of this view implies that we should probe the *memory* for the result (solution) obtained through coevolution, not the coevolving population. In the standard coevolutionary algorithm, the coevolving population is expected both to perform search *and* represent the result of the coevolution. But, there is no reason to believe that these two tasks are orthogonal, let alone mutually supportive; indeed, there is evidence that these tasks may interfere with each other [3].

A well-designed memory mechanism relieves the coevolving population from the burden (or, at least, the requirement) of representing the solution, allowing the population to concentrate on search (to improve the solution represented by the memory). The memory mechanism we introduce in this paper uses game theory, particularly Nash equilibrium, as an organizing principle. A important feature of the Nash concept is that it allows a solution to be a *collective* of strategies. The appeal of this property is highlighted by results of Nolfi and Floreano [7]. In the presence of intransitivity, they emphasize that the real solution they obtain through coevolution is not a single, objectively best champion strategy—none exists to find—but, rather a set of locations in genotype space that are optimally poised for easy transformation into alternative strategies that have been effective in evolutionary history. That is, when a population cannot simultaneously and stably represent all the strategies in certain intransitive cycles, the

best it can do is optimally traverse the cycle. This lack of a best pure strategy, coupled with a dynamic that focuses on a particular set of pure strategies, is highly suggestive of the *mixtures* that the Nash solution concept supports.

5 Construction and Operation of Nash Memory

5.1 General Framework and Instantiation

The Nash memory mechanism we examine in this paper is a particular instantiation of a general framework, which consists of two mutually exclusive sets of pure strategies, $\mathcal{N}$ and $\mathcal{M}$. In addition to the memory mechanism, we assume the existence of some external search heuristic $\mathcal{H}$. The set $\mathcal{N}$ is unbounded in size and defined to be the support set for a mixture that is secure at least against the elements of $\mathcal{N}$ and $\mathcal{M}$; that is, $\mathrm{S}(\mathcal{N}) \supseteq \mathcal{N} \cup \mathcal{M}$. (We use $\mathcal{N}$ to denote either the mixture's support set or the mixed strategy itself, as convenient.) The objective of $\mathcal{N}$ is to represent a mixed strategy that is optimal (secure) with respect to what the search heuristic has discovered thus far; we wish the protection afforded by $\mathcal{N}$ (ideally) to increase monotonically as search progresses, thereby forming a better and better approximation of a Nash strategy for the game $\mathbf{G}$. The purpose of set $\mathcal{M}$ is to act as an accumulator or memory; it contains pure strategies that are not currently useful for $\mathcal{N}$ but were in the past and may be again in the future. The capacity of $\mathcal{M}$, however, is finite. In the present realization of the abstract memory model, we define $\mathcal{M}$ to be simply an unordered set of pure strategies with a cardinality no greater than some specified value c, which gives the *capacity* of the memory. We can imagine that c can vary over evolutionary time, in some adaptive fashion, but we instead assume a fixed value.

5.2 Initialization and First Update

We initialize $\mathcal{N}$ and $\mathcal{M}$ to be the empty set. Let $\mathcal{Q}$ be a set of strategies delivered by the search heuristic to the memory mechanism; we assume $|\mathcal{Q}| < c$. The first set $\mathcal{Q}$ to arrive updates $\mathcal{N}$ such that $\mathrm{C}(\mathcal{N}) \subseteq \mathcal{Q}$ and $\mathrm{S}(\mathcal{N}) \supseteq \mathcal{Q}$; the set $\mathcal{M}$ is updated to contain those elements of $\mathcal{Q}$ not in support of $\mathcal{N}$. For subsequent values of $\mathcal{Q}$, we must test the elements of $\mathcal{Q}$ against $\mathcal{N}$ to see if $\mathcal{N}$ and $\mathcal{M}$ require updating.

5.3 Testing $\mathcal{N}$

To verify that a mixed strategy is Nash, one need only check that the mixed strategy is secure against all pure strategies; testing against all possible mixtures (of which there is an uncountable infinity) is not required [12]. Consequently, if $\mathrm{E}(q, \mathcal{N}) > 0$ for any $q \in \mathcal{Q}$, then $\mathcal{N}$ is demonstrably not a Nash strategy and we attempt to improve our approximation; otherwise, we leave $\mathcal{N}$ and $\mathcal{M}$ undisturbed and wait for the search heuristic to deliver a new set of strategies. We compute the value $\mathrm{E}(q, \mathcal{N})$ by playing q against each strategy in support of $\mathcal{N}$ and taking a weighted average of outcomes, the weights being the probability distribution for the mixture: $\mathrm{E}(q, \mathcal{N}) = \sum_{n \in \mathrm{C}(\mathcal{N})} \Pr(n|\mathcal{N})\mathrm{E}(q, n)$.

5.4 Updating $\mathcal{N}$ and $\mathcal{M}$

We define the set $\mathcal{W} = \{q \in \mathcal{Q} : \mathrm{E}(q, \mathcal{N}) > 0\}$, that is, the "winners" from $\mathcal{Q}$. Given the pre-update values of $\mathcal{N}$ and $\mathcal{M}$, we define the post-update value $\mathcal{N}'$ such that $\mathrm{C}(\mathcal{N}') \subseteq (\mathcal{W} \cup \mathcal{N} \cup \mathcal{M})$ and $\mathrm{S}(\mathcal{N}') \supseteq (\mathcal{W} \cup \mathcal{N} \cup \mathcal{M})$; we obtain the value of $\mathcal{N}'$ with *linear programming*, which is the standard method for solving zero-sum games and for which polynomial-time algorithms exist [12]. Note that $\mathrm{S}(\mathcal{N}')$ is not necessarily a superset of $\mathrm{S}(\mathcal{N})$.

As Figure 1 illustrates, the post-update value $\mathcal{M}'$ contains zero or more items from each of three sources; some strategies in $\mathcal{W}$ may not be required, some strategies in $\mathcal{N}$ may be released, and some strategies may be retained from $\mathcal{M}$ (while others may be recalled from $\mathcal{M}$ to $\mathcal{N}'$). If the resultant $\mathcal{M}'$ has a cardinality $|\mathcal{M}'| > c$, then we discard items from $\mathcal{M}'$ until the capacity constraint is met. We can imagine a number of policies for removing items from $\mathcal{M}'$; we currently remove items at random first from those retained from $\mathcal{M}$, then those released from $\mathcal{N}$ (there is never the need to remove those from $\mathcal{W}$).

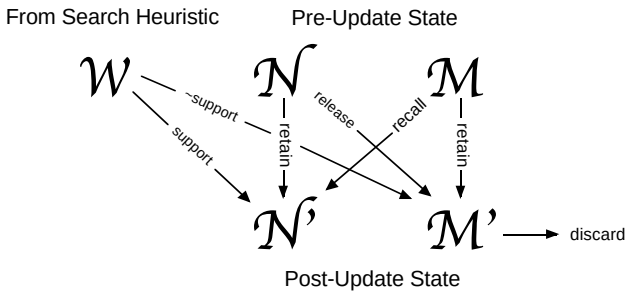

Fig. 1. Schematic of update step.

6 Intransitive Numbers Game

Watson and Pollack's *intransitive numbers game* [13] is a coevolutionary domain that is permeated by intransitive cycles. The game's geometric nature allows easy visualization of coevolutionary dynamics; we will use it to illustrate the operation of our Nash memory mechanism.

6.1 Definition

Each pure strategy of the intransitive numbers game is an n-dimensional vector, or point in n-dimensional space. For a pair of strategies α and β, the winning strategy is the one with higher magnitude in the dimension of least difference between the two strategies. More precisely, for two strategies α and β, the expected outcome $\mathrm{E}(\alpha, \beta)$ is:

$$\mathrm{E}(\alpha, \beta) = \begin{cases} 0 & \text{if } \min(h) = \infty \\ \text{sign}(\sum_{i=1}^{n} g_i) & \text{otherwise} \end{cases} \quad (1)$$

$$g_i = \begin{cases} \alpha_i - \beta_i & \text{if } h_i = \min(h) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$h_i = \begin{cases} |\alpha_i - \beta_i| & \text{if } |\alpha_i - \beta_i| \geq \epsilon \\ \infty & \text{otherwise} \end{cases} \quad (3)$$

where ϵ is the smallest magnitude difference we wish to consider significant.

Equation 1 eliminates two asymmetries that exist in the original definition found in [13]: First, the game is now formally symmetric; second, no single dimension is the arbiter when the deltas in multiple dimensions are minimal. Note also that, while we use the term "expected outcome," the game is deterministic.

While Equation 1 defines the outcome between two specified strategies, it does not define the universe of strategies we are to consider, leaving the game's definition incomplete. Clearly, the strategy space may be finite or infinite, countable or uncountable; indeed, the intransitive numbers game can provide an impoverished form of open-endedness, if desired. For our purposes, we define the strategy space to be n-dimensional vectors of natural numbers, where each dimension spans the interval $[0, k]$; this yields $(k+1)^n$ distinct pure strategies.

6.2 Game Properties

The game defined above has a single Nash equilibrium strategy for all $\epsilon < k$ ($\epsilon = k$ implies that all strategies tie each other, making them all Nash strategies—a particularly uninteresting game). The Nash strategy is the pure strategy with value k in all n dimensions, i.e., $\langle \overbrace{k, \ldots, k}^{n} \rangle$. Of course, other definitions of the strategy space may yield multiple Nash as well as mixed Nash strategies.

As with all Nash equilibria in symmetric zero-sum games, the Nash strategy of our game does not lose to any other strategy. With $\epsilon = 1$, the Nash ties only itself and beats all others; thus, the Nash also dominates all other strategies. Here, the solution concepts of Nash equilibrium and domination agree.

With $\epsilon = 0$, however, the solution concepts point to very different outcomes. In this case, the Nash additionally ties all strategies that have the value k in any dimension. (The strategies that tie the Nash are those on the surface of the n-dimensional space. Thus, as n grows, the percentage of tieing strategies asymptotically approaches 100%, even though the number of losing (i.e., interior) strategies grows exponentially. Nevertheless, for $k = 100$ and $n = 10$, the percentage of tieing strategies is still $< 10\%$. Thus, concern over the asymptotic growth of tieing strategies must keep in mind the values of k and n.)

Thus, setting $\epsilon = 0$ introduces additional ways in which a tie may occur between strategies. In particular, the Nash now ties strategies that others can beat; for example, in two dimensions, $\langle k, k \rangle$ (the Nash) ties both $\langle k, k-1 \rangle$ and $\langle k-2, k \rangle$, yet $\langle k-2, k \rangle$ beats $\langle k, k-1 \rangle$. *Therefore, for a non-Nash strategy*

to transform itself into the Nash, it may be required to discard certain skills—the ability to beat certain strategies; the goal of security is not always served by mere accumulation of prowess against opponents. Due to these additional ties, virtually no strategy is dominated; those that are dominated are done so by the Nash. With $n = 2$, only four strategies are dominated, regardless of the value of k: $\langle 0, 0\rangle$, $\langle k-1, k\rangle$, $\langle k, k-1\rangle$, and $\langle k-1, k-1\rangle$. With $n > 2$, the number of dominated strategies is only one (strategy $\langle 0, 0, \ldots, 0\rangle$). Finally, with either value of epsilon, the Nash strategy is the pure strategy with the most wins (though, this is not a general feature of Nash equilibrium strategies).

7 Experiments

Rather than simply compare the performance of a coevolutionary algorithm with and without the aid of our memory mechanism, we wish instead to probe the ability of the memory mechanism to discover mixed strategies that provide greater and greater security. To accomplish this, we use $\mathcal{N}$ as a quasi-static evaluation function against which we repeatedly evolve a population of strategies.

7.1 Methods

The following procedure essentially transforms a *co*-evolutionary domain into an evolutionary domain. We use the game defined in Section 6.1 with $\epsilon = 0$, $k = 100$, and $n = 2$. We represent an individual pure strategy for this game with a bit-string b of length 200, such that the expressed strategy is $\langle \sum_{i=0}^{99} b_i, \sum_{i=100}^{199} b_i \rangle$. The only variation operator we apply is bit-wise mutation with a per-bit mutation probability of 0.01; recombinational operators are not used. We construct a random strategy by setting each of the 200 bits to one with a probability of 0.5; this yields an expected strategy of $\langle 50, 50\rangle$. The memory's capacity is $c = 100$.

The evolution proceeds in *epochs*, as follows: 0) Initialize memory and update with a random strategy; 1) Initialize population (of size 100) with random strategies; 2) Evolve population against $\mathcal{N}$ for 30 generations (one epoch); 3) If highest-scoring individual $\hat{s}$ in population beats $\mathcal{N}$, then update memory with $\hat{s}$; 4) Goto Step 1. Each generation in Step 2 proceeds as follows: a) Evaluate each individual (pure strategy) s in the population against $\mathcal{N}$ (the score w_i obtained by each individual i is in the range $[-1, 1]$); b) The fitness of individual i is $f_i = w_i - \min(w) + 0.1$ (all fitness values are > 0); c) Copy the 10 most-fit individuals to next generation (elitism); d) Fill the remaining 90 positions with offspring using "fitness-proportionate" selection and asexual reproduction.

7.2 Results: Nash Memory

Figure 2 (Left) shows the mean and median scores obtained by the most fit individual $\hat{s}$ at the end of each epoch over 52 trials of our experiment. At the beginning of the experiment, the evolution step (Step 2, above) easily finds pure strategies that obtain the maximal score of 1.0 when played against the Nash

memory's mixture $\mathcal{N}$. Over the next 50 epochs, however, the ability of evolution to discover pure strategies that score well against $\mathcal{N}$ is gradually neutralized as the memory mechanism integrates knowledge gained from previous epochs. Indeed, over subsequent epochs, the median score obtained by evolution asymptotically approaches zero, which is the *value* of the intransitive numbers game (and zero-sum games, in general—see Section 2, above). The mean score, however, actually becomes negative; thus, the distribution is not normal. This indicates that $\mathcal{N}$ tends to be somewhat superior to the strategies that evolution is able to discover. Over the course of an experiment, the number of pure strategies in support of $\mathcal{N}$ tends to grow. The mean size of $\mathrm{C}(\mathcal{N})$ at Epoch 500 is about 45, though the distribution is not normal (median is 50).

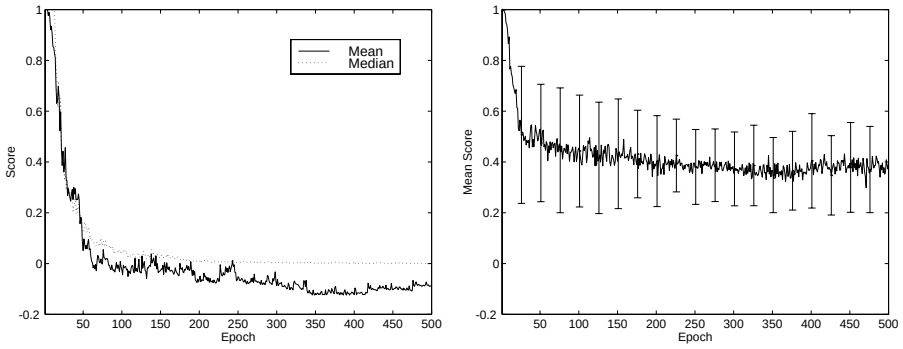

Fig. 2. Performance of evolution over time against (Left) Nash memory, over 52 trials, and (Right) BOG memory ($m = \infty,\ l = 100$), over 54 trials.

Figure 3 shows the behavior of a typical trial. (For convenience, $\mathcal{N}^i$ will denote the value of $\mathcal{N}$ at the beginning of the i-th epoch, similarly for $\mathcal{M}^i$.) The memory's strategy $\mathcal{N}^1$ is initialized to the pure strategy $\langle 53, 41\rangle$ (indicated by the square in Figure 3, Right). The evolution step (Step 2, above) easily finds a pure strategy $\langle 55, 47\rangle$ that beats $\mathcal{N}^1$. Accordingly, we update the memory such that $\mathcal{N}^2$ is $\langle 55, 47\rangle$, and move $\langle 53, 41\rangle$ to the memory set $\mathcal{M}^2$.

In Epoch 2, the evolution step discovers the pure strategy $\langle 48, 52\rangle$, which beats $\mathcal{N}^2$. Since $\mathcal{M}^2$ contains something, we must evaluate the performance of $\langle 48, 52\rangle$ against the contents of $\mathcal{M}^2$ before we can determine $\mathcal{N}^3$. Though $\langle 48, 52\rangle$ beats $\langle 55, 47\rangle$, we find that it loses to $\langle 53, 41\rangle$ (the strategy in $\mathcal{M}^2$). Thus, we find an intransitive cycle; indeed, this intransitivity is symmetric (as in the Rock-Scissors-Paper game), so no one pure strategy can be argued to be any better than another (in any sense). This is not to say that the three strategies are interchangeable, however, for each one has unique strengths and weaknesses with respect to the other two. Since none of these pure strategies has an empty vulnerability set (see Section 3, above), $\mathcal{N}^3$ must be a mixture of some kind. The solution (found with linear programming) happens to be a mixed strategy

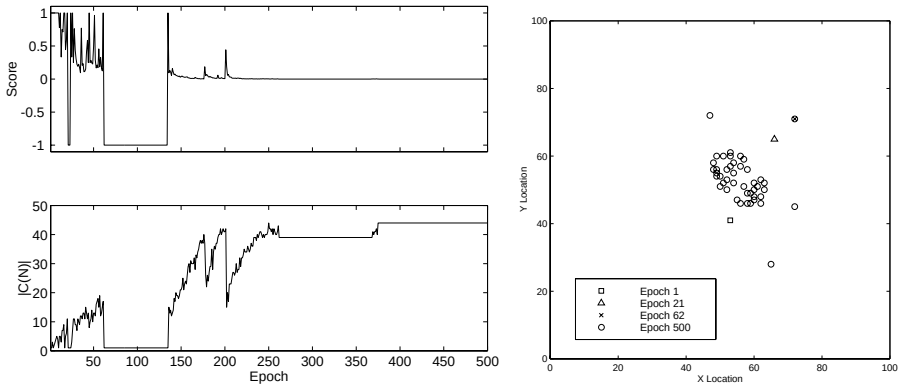

Fig. 3. Example run of evolution against Nash memory over time. Left: Upper graph gives maximal score achieved by evolution against $\mathcal{N}$ at the end of each epoch; lower graph shows size of support set $\mathrm{C}(\mathcal{N})$ at the beginning of each epoch. Right: Pure strategies in support of $\mathcal{N}$ at the beginning of Epochs 1, 21, 62, and 500.

where each of the pure strategies is played with equal probability; $\mathcal{N}^3$ has all three strategies in support and $\mathcal{M}^3$ is empty.

By the time we arrive at $\mathcal{N}^{20}$, eleven strategies are in support. In this epoch, evolution discovers pure strategy $\langle 66, 65\rangle$ (triangle in Figure 3, Right), which beats each of the eleven support strategies. Further, because $\mathcal{N}^{21} = \{\langle 66, 65\rangle\}$, we can infer that $\langle 66, 65\rangle$ is also secure against the contents of $\mathcal{M}^{20}$. The strategy $\mathcal{N}^{21}$ is sufficiently good that the evolution step obtains the worst possible score (-1) over the next few epochs. Later, we encounter a similar, but more pronounced, situation where $\mathcal{N}^{62}$ is the pure strategy $\langle 72, 71\rangle$ ('x' in Figure 3, Right). This strategy remains until Epoch 136, where it is replaced by a mixture. The mixture $\mathcal{N}^{500}$ (circles in Figure 3, Right) retains $\langle 72, 71\rangle$ in support.

7.3 Results: Best of Generation and Dominance Tournament

We also run our experiment using the best-of-generation memory mechanism with $m = \infty$, $l = 100$ (see Section 4, above). During evaluation, an evolving individual's score against the memory is the average score obtained against the strategies sampled from the memory; the highest-scoring individual at the end of an epoch is added to the memory if its score is > 0. Figure 2 (Right) shows the mean scores (with standard deviation) obtained by the most fit individual $\hat{s}$ at the end of each epoch over 54 trials of this experiment; these data are normally distributed. The mean score obtained by $\hat{s}$ between epochs 250 and 500 is ≈ 0.375 ($\sigma \approx 0.160$). In other experiments where $m = 100$, $l = 100$ (not shown), the expected score of $\hat{s}$ at steady-state is ≈ 0.758 ($\sigma \approx 0.148$). Thus, the organizing principle behind BOG provides considerably less effective learning.

We next use the Dominance Tournament as our memory mechanism. The score given to an evolving individual is the number of consecutive pure strate-

gies, from the most to least recently placed in the memory, that the individual beats. At the end of an epoch, if any individuals exist that beat the entire contents of the memory, we randomly select one and place it in the memory. The DT method quickly converges onto a pure strategy that cannot be dominated by any other *with respect to the memory's contents*. Because we only consider those strategies in the memory to determine dominance, and not the entire universe of strategies, false positives may occur; indeed, all but four strategies in our game are actually *non*-dominated. Thus, the strategy onto which we converge is highly path-dependent; particularly, we may converge onto the dominated strategy $\langle 100, 99 \rangle$ (or $\langle 99, 100 \rangle$—see Section 6), after which the memory will reject the Nash $\langle 100, 100 \rangle$, which dominates $\langle 100, 99 \rangle$ but does not beat it. Further, of all the pure strategies onto which we may converge, only the Nash equilibrium strategy has an empty vulnerability set; all the others are beatable.

7.4 Results: Bootstrapping with the Nash Memory

The results we report in Section 7.2 only show that the Nash memory is able to learn a mixed strategy $\mathcal{N}$ that is secure against what the evolution step is likely to discover, given the limitations we place on the evolution, such as the random initial population. In particular, the mixed strategies $\mathcal{N}$ obtained above are *not* the Nash equilibrium for the numbers game.

Therefore, we create a new outer loop around the method described in Section 7.1 to explore the ability of the memory to constructively contribute genetic material to the search process. When the evolution step is unable to score higher than 0.04 against $\mathcal{N}$ for 50 consecutive epochs, we change the procedure used to initialize the population: A snapshot of $\mathcal{N}$ is taken and used to initialize populations in all subsequent epochs (until our outer-loop criterion is met once again). Specifically, the initial population contains the strategies in support of $\mathcal{N}$ in proportion to their probabilities in the mixture distribution.

This new initialization process allows the evolution step to further challenge the Nash memory. While we do not converge onto the precise Nash strategy for our game (which is $\langle 100, 100 \rangle$), we do obtain a mixture $\mathcal{N}$ (with 123 pure strategies in support) that virtually eliminates vulnerability to all strategies ($\mathcal{N}$'s worst score ≈ -0.05) except the true Nash ($\mathcal{N}$'s score ≈ -0.323); the memory mechanism is poised to accept the true Nash strategy, if search can find it. Thus, the solution $\mathcal{N}$ is an excellent approximation to the true Nash strategy *in behavior*, though superficially they appear nothing alike.

8 Conclusion

We examine the performance of three distinct memory mechanisms using Watson and Pollack's intransitive numbers game: Nash memory, Best-of-Generation, and Dominance Tournament. We do not intend to argue that the BOG and DT mechanisms cannot improve the performance of a coevolutionary algorithm; indeed, the BOG approach is known to help, e.g. [9, 7]. We are more interested in the role of coevolutionary memory and its organization.

We show that the Nash memory improves its collection of traits, expressed as a mixed strategy, as search exposes the memory to new traits. Over time, the mixed strategy asymptotically approaches the performance of the Nash equilibrium strategy for the numbers game, thus providing 1) an excellent approximation of the (game-theoretic) solution, and 2) a gradually increasing challenge for evolving strategies. In contrast, the BOG method exhibits only limited ability, when faced with pervasive intransitivity, to increasingly challenge evolution; lacking a strong organizing principle, BOG does not converge to the Nash equilibrium. The DT method implements the principle of domination, but only with respect to local knowledge; without the global knowledge required to properly determine domination, the numbers game easily leads the DT method astray.

Acknowledgements

The authors wish to thank Anthony Bucci, Edwin de Jong, Shivakumar Viswanathan, and Richard Watson for many useful discussions.

References

1. R. Brooks and P. Maes, editors. *Proc. 4th Conf. on Artif. Life*. MIT Press, 1994.
2. D. Cliff and G. F. Miller. Tracking the red queen: Measurments of adaptive progress in co-evolutionary simulations. In F. Moran et al., editors, *3rd Euro. Conf. on Artificial Life*, pages 200–218. Springer Verlag, 1995.
3. S. G. Ficici, O. Melnik, and J. B. Pollack. A game-theoretic investigation of selection methods used in evolutionary algorithms. In A. Zalzala et al., editors, *Proc. of 2000 Congress on Evolutionary Computation*, pages 880–887. IEEE Press, 2000.
4. S. G. Ficici and J. B. Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In C. Adami et al., editors, *Proc. of the Sixth Conf. on Artificial Life*, pages 238–247. MIT Press, 1998.
5. D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1998.
6. D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, 1989.
7. S. Nolfi and D. Floreano. Co-evolving predator and prey robots: Do 'arm races' arise in artificial evolution? *Artificial Life*, 4(4):311–335, 1998.
8. C. W. Reynolds. Competition, coevolution and the game of tag. In Brooks and Maes [1], pages 59–69.
9. C. Rosin and R. Belew. New methods for competitive co-evolution. *Evolutionary Computation*, 5(1):1–29, 1997.
10. K. Sims. Evolving 3d morphology and behavior by competition. In Brooks and Maes [1], pages 28–39.
11. K. O. Stanley and R. Miikkulainen. The dominance tournament method of monitoring progress in coevolution. In A. Barry, editor, *2002 Genetic and Evolutionary Computation Conference Workshop Program*, pages 242–248, 2002.
12. P. R. Thie. *An Introduction to Linear Programming and Game Theory*. John Wiley and Sons, 1988.
13. R. A. Watson and J. B. Pollack. Coevolutionary dynamics in a minimal substrate. In L. Spector et al., editors, *Proc. 2001 Genetic and Evolutionary Computation Conf.*, pages 702–709. Morgan Kaufmann, 2001.