Challenges in Coevolutionary Learning: Arms-Race Dynamics, Open-Endedness, and Mediocre Stable States

Sevan G. Ficici and **Jordan B. Pollack**
DEMO Lab
Computer Science Department
Volen National Center for Complex Systems
Brandeis University
Waltham, Massachusetts USA
http://www.demo.cs.brandeis.edu

Abstract

Coevolution has been proposed as a way to evolve a learner and a learning environment simultaneously such that open-ended progress arises naturally, via a competitive arms race, with minimal inductive bias. Nevertheless, the conditions necessary to initiate and sustain arms-race dynamics are not well understood; mediocre stable states frequently result from learning through self-play (Angeline & Pollack 1994), while analysis usually requires closed domains with known optima, like sorting-networks (Hillis 1991). While intuitions regarding what enables successful coevolution abound, none have been methodically tested. We present a game that affords such methodical investigation. A population of deterministic string generators is coevolved with two populations of string predictors, one "friendly" and one "hostile"; generators are rewarded to behave in a manner that is simultaneously predictable to the friendly predictors and unpredictable to the hostile predictors. This game design allows us to employ information theory to provide rigorous characterizations of agent behavior and coevolutionary progress. Further, we can craft agents of known ability and environments of known difficulty, and thus precisely frame questions regarding learnability. Our results show that subtle changes to the game determine whether it is open-ended, and profoundly affect the existence and nature of an arms race.

Introduction

Most machine learning (ML) systems operate by optimizing to a fixed fitness function, or learning environment, and typically require considerable inductive bias in order to succeed; this inductive bias takes the form of either a learner that is *pre-adapted* to the learning environment, or a carefully *gradient-engineered* fitness landscape that provides the learner with a clear path towards a global optimum. In both cases, however, the onus inevitably falls upon the human user of ML technology to imbue the learning system with the appropriate bias. Thus, results are often attributable to inductive bias as much as, or more than, the ML methods used. As learning domains become more intricate and demanding of ML systems, however, both methods of bias engineering quickly become infeasible: gradient engineering turns overwhelmingly complex, and, following the observation that "you can only learn what you almost already know," pre-adaptation requires the learning problem to be already substantially solved.

To address these problems, *coevolution* has been proposed as a way to evolve a learner and learning environment simultaneously such that progress arises naturally with minimal inductive bias. In coevolution, however, the terms 'learner' and 'environment' no longer denote absolute roles, but relative ones; each participant in a coevolutionary system is both a learner as well as an environment against which other participants learn — the conventional asymmetry between learner and environment does not exist.

The key to successful coevolutionary learning is a *competitive arms race* between opposed participants. Competitors must be sufficiently well-matched in skill to force each other to improve. The difference between what participants already know and what they must learn is critical: if one competitor becomes relatively expert such that the opponent is "overpowered," then the opponent will fail to find a gradient towards improvement and be subsequently unable to offer continued challenge, thereby breaking the arms race. If a balance in the arms race is maintained, on the other hand, coevolution is hypothesized to provide a way to gradually evolve opposing forces such that each is always suitably pre-adapted to learn against the other while, at the same time, offering a suitably engineered gradient against which the other can learn. In *open-ended* domains, coevolutionary progress can, theoretically, continue indefinitely.

Nevertheless, the precise conditions necessary to initiate and sustain such arms-race dynamics are neither definitively known nor well understood; *mediocre stable-states* (MSS) (Angeline & Pollack 1994; Pollack, Blair, & Land 1997) are a common result in coevolutionary systems, where the agents in the evolving population(s), to anthropomorphise, discover a way to *collude* to give the impression of competition without actually forcing each other to improve in any "objective" sense. This phenomenon is analogous to that found in accounts of World-War I trench warfare (Axelrod 1984), where opposing forces established ritualized acts of aggression meant to appear genuine to their respective commanders that were, nevertheless, completely predictable to each other, and thus of no real threat.

Complicating research into the arms-race mechanism is the fact that analysis of coevolutionary systems usually requires domains with known optima, like sorting-networks (Hillis 1991), and simple differential games (Isaacs 1965), so that an objective metric of performance is available. Unfortunately, these domains are closed-ended, and are thus categorically less interesting than open-ended domains. Without quantitative metrics of agent behavior, researchers in open-ended coevolutionary domains can do no better than use qualitative language to describe agent behavior and system progress. Indeed, this problem has been recognized by researchers in the pursuer-evader domain (Cliff & Miller 1996).

Thus, while current research is rich with insights and intuitions regarding what enables coevolution, there is, at the same time, a paucity of domains that can serve as systematic testbeds for these intuitions; we present a game that affords such methodical investigation. Our game involves three agents: one bitstring generator, and two string predictors — one "friendly" and one "hostile"; the generator is to behave in a manner that is simultaneously predictable to the friendly predictor partner yet unpredictable to the hostile predictor opponent. The two predictor roles produce a tension between cooperative and competitive pressures, respectively. Because agent behavior is expressed as a binary time series, we can use information theory to quantitatively assess agent behavior and coevolutionary progress. Further, we are able to hand-build agents of known ability, which implies that we can also build environments of known difficulty. We may thus pose precisely-framed questions regarding learnability, arms-race dynamics, mediocre stable-states, and open-endedness.

Our results demonstrate the expressiveness of our domain in investigating coevolution; many different dynamics can be produced by simple changes to our game. While our substrate is capable of representing both good generators and predictors, we find that high-quality players are not an inevitable outcome of coevolution; the obvious competitive approach to coevolution in our game (one that omits the friendly partner) does not produce an open-ended arms-race. Rather, a mediocre stable-state or closed-ended system is the result, depending on a seemingly minor change in how the game is scored. Mediocre stable-states result from a variety of causes in coevolutionary research. Due substantially to our rigorous metric of behavior, we can refine the notion of MSS and begin a taxonomy of such causes. All three players in our game are found required to enable an arms-race. The viability of an arms race relies on the sustained *learnability* (Pollack, Blair, & Land 1997) of environments; we are able to construct environments that are too easy and too difficult for learning to take place, and quantitatively demonstrate the poorness of these environments.

This paper is organized as follows: we first explain our game in detail, discuss the recurrent artificial neural network substrate used for our experiments, and describe the evolutionary algorithm. Next, key concepts from information theory that are relevant to this work are introduced. These concepts are then integrated into the framework of arms-race dynamics. Results are presented and analyzed. Finally, we summarize our work and present concluding remarks.

Game Setup, Substrate, and Evolutionary Algorithm

Illustrated in Figure 1, our game is played by three agents that operate in discrete time and space. The generator, $\mathcal{G}$, is an agent that ballistically produces a binary time series. That is, its behavior is determined solely by its own internal dynamics; at each time step, the generator simply outputs a single new bit. The predictors, $\mathcal{F}$ and $\mathcal{H}$, are agents that simultaneously try to predict the generator's output for the current time step; given their own internal state and the generator's output from the previous time step as input, the predictors also output a single bit in synchrony with the generator. The generator's job is to behave in a manner that is both predictable to the friendly predictor, $\mathcal{F}$, and unpredictable to the hostile predictor, $\mathcal{H}$. The purpose of having both friendly and hostile predictors in our game is to explore how the opposed needs for predictability and unpredictability affect coevolutionary dynamics. Each match lasts one thousand time steps.

Agents are coevolved in three distinct populations, one population for each role (represented by $\mathcal{F}$, $\mathcal{G}$, and $\mathcal{H}$) in our game. The three populations are all of a fixed size of 75 agents. For each generation of evolution, all generators are played against all friendly and hostile predictors. Agent performance is measured strictly in terms of the number of correct and incorrect predictions made. Scores across all games are averaged to derive fitness values. Game scores for all agents range between [0, 1]. The exact formulas used for scoring predictors are discussed below in the experiment descriptions. A generator's score is computed by subtracting the average score of its hostile opponents from the average score of its friendly partners and normalizing the result to fall within the range [0, 1]; values above 0.5 thus indicate that a generator is able to make itself more predictable to friendly predictors than hostile ones.

The substrate used for the agents is an enhanced version of the deterministic, discrete-time recurrent artificial neural network used in the *GNARL* system (Angeline, Saunders, & Pollack 1994); the network enhancement consists of a set of nine new transfer functions, *min*,

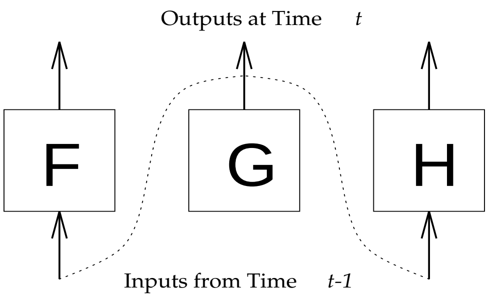

Figure 1: Game Setup.

max, *sum*, *product*, *sig-prod* (sigmoid applied to product), *unit-time-delay*, *sign*, *uni-linear-truncate* (truncate outside range [-1, 1]), *dec-linear-truncate* (truncate outside range [-10, 10]), in addition to the traditional sigmoid function. Though these supplementary transfer functions increase the range of behavior significantly, they by no means guarantee successful coevolution, as we will see. Unfortunately, the new functions also make network analysis much more difficult.

GNARL is used to coevolve networks' internal architectures and weights. Generators are allowed to have as many as 60 hidden nodes and 400 weights. Because the prediction task is more difficult, predictors are allowed to have as many as 150 hidden nodes and 700 weights. These values merely reflect intuition and are not known to be optimal; indeed, actual network sizes fall far below these limits.

Mutation is the only genetic operator used by the GNARL algorithm — crossover is not used. Five forms of mutation are implemented: *change-weights*, *add-hidden-nodes*, *remove-hidden-nodes*, *add-weights* (connections), and *remove-weights* (connections). When a hidden node is removed, all efferent and afferent connections from and to that node are removed as well. New nodes are added with no connections. Network weights are modified by adding a gaussian to each weight. Only a single form of mutation is applied to a network when it is modified. The overall severity of mutation performed to a network is determined by its *temperature*, which is an inverse function of its fitness. The higher the temperature of a network, the more severe the mutation will be.

The input and output layers of the networks are fixed. Generators and predictors have a single, real-valued output that is thresholded to a binary value. Though the game formally defines predictors to have a single input, our current experiments provide predictors with a small buffer to enhance performance: the predictors have five binary-valued inputs, corresponding to the last five outputs of the generator at times $t-1, ..., t-5$. Note that this predictor enhancement in no way obviates the need for a recurrent network architecture: generator behavior can be induced only by observation over time.

Metric of Behavior

Introduction

In this section we introduce two key notions from the field of information theory that provide our game with a rigorous and quantitative metric of agent and system behavior, namely *entropy* and *order*. Rather than give their formal mathematical definitions, we emphasize a more intuitive explanation of these concepts and their implications as they relate to our domain. Readers interested in more formal detail are referred to (Hamming 1980).

Entropy

Information theory is concerned with characterizing signals and their transmission. A signal *source* produces some symbol, which is passed through a channel to a *receiver*. We assume, for our purposes, that the channel does not distort the signal. The *entropy*, h, of a source reflects the receiver's uncertainty as to what it will receive. The higher the entropy, the less certain the receiver is, and the more it learns once the symbol is actually received. Thus, entropy is a measure of the amount of *information* in a signal. More precisely, the entropy of a source is equal to the average number of bits of information produced (conveyed) per generated symbol.

By indicating the uncertainty of the receiver, entropy inversely indicates the degree to which the source can be *predicted* by the receiver, that is, the receiver's certainty. We must be careful to point out that the receiver's opinion of what the next symbol will be is based exclusively upon the *observed behavior* of the source — assumptions about the source's internal dynamics are not made.

Order

If the receiver's certainty is based upon observation of the source, we can ask "How much observation is required to maximize the receiver's certainty?" For example, let us consider some binary source, $\mathcal{S}$. If the receiver only tallies the number of occurrences of *0* and *1*, this source may be found to produce each 50% of the time. With this amount of behavioral context, the receiver's certainty is zero and entropy is measured at $h = 1.0$. Nevertheless, it may be that if the receiver keeps track of the previous symbol received, then the source will be found simply to be alternating between *0* and *1*; in this case, a behavioral context of one symbol makes the source completely predictable. Measured entropy would now be $h = 0.0$. If the receiver keeps track of yet another symbol, now the previous two, no additional advantage is gained with respect to source $\mathcal{S}$.

The minimal amount of behavioral context needed to maximize a receiver's certainty of a source is the *order* of the source. The order is equal to the number of symbols that must be tracked, that is, the size of the history *window* needed to maximize receiver certainty. The entropy measured when using a window size equal to a source's order is the *true entropy* of the source; window sizes larger than a source's order will produce measurements equal to the source's true entropy, but not lower. Thus, a receiver cannot increase its certainty of a source by using a window size larger than the source's order.

Order Statistics and Measured Entropy

With our example source, $\mathcal{S}$, above, we first measured entropy without keeping track of the previously generated symbol; this is equivalent to measuring entropy with a window size of zero, or, alternatively, measuring entropy with zero-order statistics. Our second measurement, then, used a window size of one, or first-order statistics. Our zero-order measurement gave us an entropy of $h = 1.0$, but the first-order measurement fell to the true entropy of $h = 0.0$. Indeed, measured entropy will always monotonically decrease as window size is increased, and eventually reach a source's true entropy, as illustrated in Figure 2.

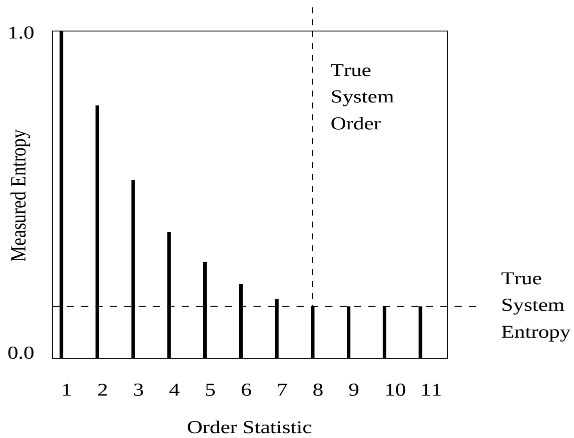

Figure 2: Measured vs. True System Entropy and Order.

A source with true entropy $h = 0.0$, such as S, is completely predictable and regular. In contrast, a binary source with maximum true entropy of $h = 1.0$ entirely lacks structural regularity and cannot be predicted better than random, on average, without specific knowledge of its internal works. For a source with true entropy somewhere in between, $0.0 < h < 1.0$, there exists both a *regular component* and an *irregular component* to the source's signal. The regular component is that portion of the signal that can be reliably predicted, while the irregular component is that portion that cannot. By definition, the information content of a source must reside exclusively in the irregular component.

Order as Complexity

System order is also equal to the logarithm of the maximal number of states required for a Markov model to reproduce behavior statistically identical to a source; entropy reflects the degree of certainty in the model's state transitions. Consider a randomly behaving binary source, with true entropy of $h = 1.0$. We find that the minimal window size needed to maximize a receiver's certainty of this source is *zero*! Since the order of such a source is zero, the equivalent Markov model requires $2^0 = 1$ state to reproduce statistically identical behavior. This result is understandable since there exists no signal structure to capture through state. This view of system complexity thus considers random sources to be simpler than completely predictable sources of higher order; the size and structure of the Markov model is what counts, not the compressibility of the produced signal (Kolen & Pollack 1994).

Where's the Arms Race?

When considered together, order and entropy form the nexus between generator complexity and predictor power: if a signal has a regular component, then that component can be predicted assuming the power of the predictor is sufficient; that is, the predictor must use an order statistic, i.e., history window, of size $m \geq n$, where n is the order of the signal being predicted. If the predictor's window size is m, such that $m < n$, then it will be able to predict only that portion of the signal's regular component that is detectable when measuring the signal's entropy with m^{th}-order statistics. Recall that as window size decreases, measured entropy increases; thus, predictors using smaller windows will necessarily be weaker than those using larger windows.

Because irregular signal components are inherently unpredictable, and our three-player game requires generators to be predictable to friendly predictors, generators must maintain substantial regular components in order to succeed. Nevertheless, generators need to be unpredictable to the hostile predictors. The only way both goals can be effectively met is for the generators and friendly predictors to evolve system order and predictive power that are closely matched, yet greater than the predictive power of the hostile predictors.

Regular signals allow for a general solution to the prediction task. A predictor of power n can predict any generator of order $m \leq n$; to escape prediction, therefore, a generator has no choice but to increase its order above n. The amount by which the generator increases its order and the unpredictability it exhibits at lower order-statistics determines how much the predictor's performance degrades. Of course, a generator may increase its true entropy instead; doing so, however, will also defeat any hopes of being predicted by the friendly predictor.

The assumption up to now has been that predictors will actually evolve such a general prediction algorithm and evolve the functional equivalent of ever-growing history windows. Of course, this represents an idealized solution; in reality, the issue of generalization vs. specialization is intimately tied to that of diversity and domain "physics." Nevertheless, having some notion of what idealized predictors can and cannot do, and what an idealized arms race looks like, provides a useful framework in which to examine empirical results.

Experiments and Results

Testing the Search Space

Our concern here is to verify that the combination of our substrate and problem domain yields a search space rich enough to support an arms race. Our first question is whether the substrate is capable of representing irregular generators and regular, high-order generators. As recurrent networks are known to produce chaotic behavior with ease, this question is partly rhetorical. Nevertheless, we seek to evolve generators in an environment that closely mimics our game. Using hand-built (non-network) predictors that employ a variety of window sizes, we evolve generators to become minimally predictable to weaker predictors while remaining maximally predictable to more powerful ones. The hand-built predictors are used as the basis of a boot-strapping process: predictors of known power are used to evolve generators of specific order complexity. Seventy generators, predominantly of orders two through eight, are thus evolved. Our second question is whether, given these generators of known order, predictors can be evolved to predict them. Evolving against this fixed population of generators, we are able to produce predictors that perform at an average rate of 75% to 85% correct prediction, or 50% to 70% better than random. The nominal range of behaviors demonstrated by our networks suggest a non-trivial solution space.

The Two Half-Games

We begin our analysis of game results by looking at the two possible *half-games*. These are versions of the game where generators coevolve only with friendly predictors or hostile predictors, but not both. The purpose of the half games is to explore convergent and competitive pressures in isolation such that we may compare and contrast results with those that include all three players.

In the first half-game, we coevolve generators with friendly predictors only. The system is asked, essentially, to establish a convention of behavior. This version of the game quickly converges in less than twenty generations. Generators and predictors alike display static behaviors of all ones or zeros, depending on which the system settles on. All agents receive perfect scores. Simply, there is no pressure for complexity and none is observed.

The second half-game, coevolving generators against hostile predictors, is more illuminating. A reasonable intuition would expect an arms race to develop between ever more complex generators and ever more powerful predictors. The actual outcome depends strongly on how the game is scored. One scoring method (A) gives predictors a point for each correct prediction. The tallies are then normalized to the percentage of correct or incorrect predictions. Another scoring method (B) tallies correct predictions, like method A, but predictors are now rewarded only to the extent that they perform better than random; predictors that get less than or equal to 50% correct prediction receive a score of zero. Method A gives maximal reward to a generator when a predictor is unable to make any correct predictions, whereas method B gives maximal reward when a predictor performs no better than random. No other experiment parameters are modified between these two scoring methods.

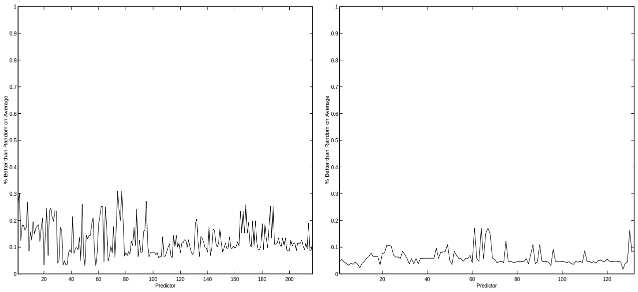

Figure 3: Performance of Best Predictors vs. Best Generators (Methods A, left, and B, right). The champion predictors are arranged along the X axis in the order in which they appear in evolutionary time. The Y axis is prediction performance as percent better than random guessing.

With each scoring method, we collect the champion predictor and generator from each generation and play the two sets of champions against each other. Predictors perform considerably worse than in the fixed-environment substrate tests described above. With method A the average prediction rate is 13% better than random (56.5% correct), though many predictors perform 20%–30% better than random, as shown in Figure 3a. Method B gives an average prediction rate of 6% better than random (53% correct); these data, shown in Figure 3b, have a standard deviation of 0.03 — less than half that of Figure 3a. The predictor scores alone are not particularly informative. We must look at the generators to discover why these scores are so low and whether they are low for the same reason.

Our principle method of measuring generator behavior is to take entropy measurements over a range of window sizes; Figures 4 and 5 graph these *entropy contours* for the best generators that arise over evolutionary time in sample runs for scoring methods A and B, respectively. These contour graphs indicate the extent to which the generators can be predicted when observing their behavior with various window sizes.

Neither figure suggests the presence of an arms race — generator characteristics do not change during the runs. The contours produced by method A drop rapidly; this indicates that the generators are substantially predictable and regular. In contrast, the contours produced by method B decline gradually and consume a much greater volume of space, indicating considerably more irregular (unpredictable) generator behavior. This difference results simply from the change in scoring method.

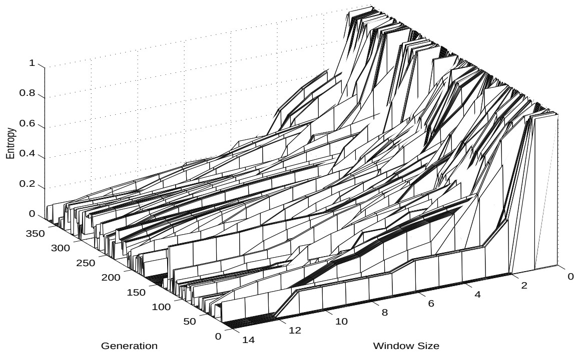

Figure 4: Generator Behavior by Scoring Method A.

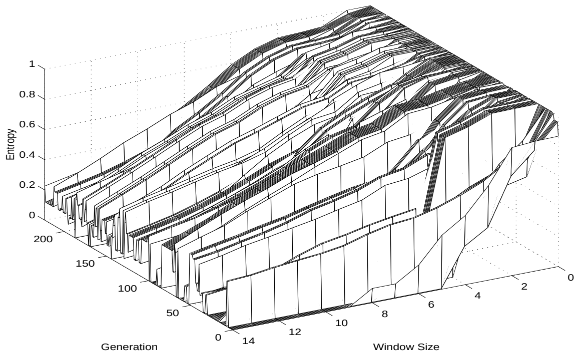

Figure 5: Generator Behavior by Scoring Method B.

Significantly, Figure 5 shows that the best generator from the initial population is already of relatively high order and entropy, while Figure 4 does not — not because such generators do not exist within the initial population of this run, but rather because they are not as adaptive to scoring method A. That low-order, regular generators out-score irregular ones by method A indicates that simple prediction strategies can be elicited from the initial predictor population — prediction strategies against which the most adaptive generators act as potent *anti-signals*, sequences that make predictors perform worse than random (Zhu & Kinzel 1997). Since the generators are ballistic, they require some homogeneity amongst the predictors in order to be viable as anti-signals. Consequently, the selected-for generators must reflect this homogeneity and thus do not provide a suitably diverse environment for subsequent predictor evolution: the predictors do evolve to counter the generators, but through specialization instead of generalization. At this point, much of the evolutionary turnover is due to exploitation of peculiar weaknesses in the agents. Rather than enter an arms race, the two populations form loose food-chain relationships and fall into a circular pattern of *convention chasing* — a mediocre stable-state.

Much like the *CIAO* graphs of (Cliff & Miller 1995), Figure 6 shows the results of playing the champion predictors against the champion generators that were evolved by scoring method A. Each position on the axes represents a moment in evolutionary time when a change in champion occurred. A column (going up) thus shows a particular predictor champion playing against all the generator champions in the order in which they reigned during the run; similarly, a row (going right) shows a particular generator champion playing against all predictor champions in the order in which they reigned. The data are thresholded such that a white point represents a prediction score of $\geq 40\%$ better than random, and black points $< 40\%$.

The important details are the many prominent horizontal and vertical lines. The pattern of a line serves to characterize an individual predictor or generator with respect to the opposing population. For example, champion generators #61, #100, and #129 have very similar (horizontal) profiles regarding which champion predictors can predict them. The gaps in these lines indicate periods where the predictor champion has lost some ability to predict this class of generator behavior. This repeated appearance and loss of particular generator and predictor behaviors is the manifestation of convention chasing.

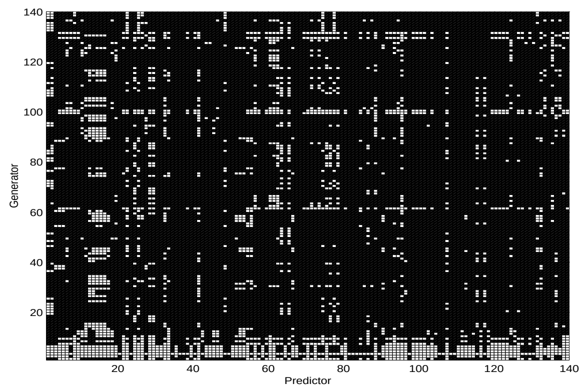

Figure 6: Evidence of Convention Chasing.

A frequently effective prediction behavior seen with scoring method A is to simply predict all ones or zeros; this strategy is general in the sense that it provides performance similar to random guessing, on average. Nevertheless, this strategy also dampens any tendency to evolve generator complexity or irregularity: because a random generator can cause it to do no worse than a simple oscillator, there is nothing to be gained by evolving or maintaining irregular generators. In contrast, irregular generator behavior is clearly adaptive by scoring method B because it guarantees a minimal predictor score, regardless of what the predictor does, short of specialized memorization. Thus, method B defines an optimal generator strategy and makes the game closed-ended. The anti-signal behavior from method A does not represent an optimal generator solution, however, because its effectiveness depends entirely upon the simplicity and homogeneity of the predictor population.

The key observations of our analysis result from being able to characterize the nature of adaptiveness with respect to a known environment, and are independent of the mechanics of evolution, due to the evolutionary algorithm (GNARL). The utility of isolating the contributions made by these two components to an evolutionary system's operation is considerable. Historically, much more attention is paid to the algorithmics than to the environment. This imbalance is perhaps due to the inherent opaqueness of most problems domains with regard to adaptiveness. Because information theory provides tools to analyze and synthesize agents, our domain allows detailed exploration of adaptiveness in isolation from the vagaries of mutation operators, reproduction schemes, and so on.

The Full Game

For the *full-game*, where all three populations participate, we keep scoring method B, to provide a pressure to evolve irregular generators. The tension between the opposed requirements of being unpredictable to hostile opponents while being predictable to friendly partners is the feature of interest in the full-game. In this case, a reasonable intuition might expect these opposing forces to stifle any possibility of an arms race; to the contrary, we believe this tension to be a necessary ingredient. Two particular runs of the full-game are discussed below; the first run displays features of an arms race, while the second run exhibits a phenomenon more typical of the full-game setup, one that likely subsumes the arms race dynamic.

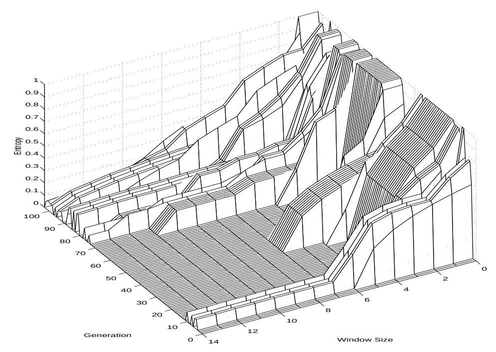

Figure 7: Generator Behavior in First Full-Game Run.

Evidence of the arms race is found first through entropy contour graphs. Figure 7, unlike Figures 4 and 5, shows a general increase in the order and entropy of the best generators over evolutionary time; that is, higher-order generators are eventually required to keep the hostile predictors at bay while remaining predictable to the friendly predictors, ostensibly because the hostile predictors have mastered simpler generator behaviors.

From each generation of the run we save the best generator, friendly predictor, and hostile predictor and play these champions against each other to look for evidence of such skill accumulation in predictors. Figure 8 shows how well hostile predictor champions perform against generator champions. The shade of the data point refers to the success of the predictor in the match; lighter shades indicate better prediction.

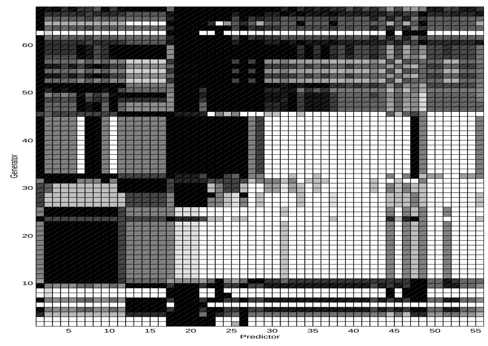

Figure 8: Champion Hostile Predictors vs. Champion Generators. Lighter shade indicates greater predictor success.

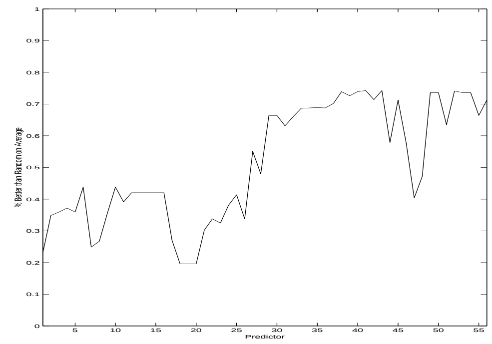

Figure 9: Average Scores of Champion Hostile Predictors against Champion Generators.

Figure 9 graphs the average performance of each champion predictor against the entire set of champion generators, that is, the average value of each column of Figure 8. Starting approximately with predictor #18, we see a clear accumulation of skill as the predictors evolve. The first group of generators to be mastered is the set #11–#25, then generators #26–#31 (starting around predictor #23), and finally generators #32–#45 (starting with predictor #27). The generators numbered #46 and higher come to be predicted, on the whole, with moderate success (up to about 50% above random).

Paradoxically, Figure 8 appears to show that predictors #1–#17 already predict a rather wide range of generators. This seems to contradict the view that predictors require evolution to generalize. If we look at Figure 9, we see that predictor #15, for example, performs no better on average than predictor #25. But, predictors #15 and #25 have very different characteristics, according to Figure 8; predictor #15 scores 40%–50% above random against most generators, whereas predictor #25 does no better than random against many, but near perfect against others. Thus, predictor #25 is specialized. Yet, predictor #15 is a generalist only in a weak sense because it does not effectively adapt its behavior to match different generators. In contrast, predictors that arise later in the run (e.g., #40) actually master a variety of generators, and exhibit a more substantive form of generalization.

In summary, these data suggest a decomposition of the arms-race notion into finer-grained events. Figures 8 and 9 suggest that predictor ability increases through accumulation of specific and distinct skills rather than a more diffuse improvement of a monolithic prediction strategy. Figure 7 allows us to see how generators evolve in response.

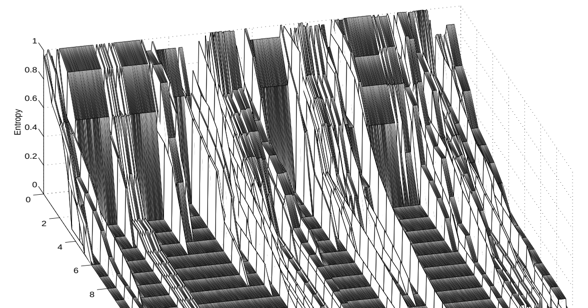

Figure 10: Punctuated Shifts in Generator Behavior from Second Full-Game Run.

More typically, however, the full-game exhibits punctuated shifts in generator behavior, from simple to complex and back again, rather than the monotonic increase in complexity indicative of an arms race. Figure 10 shows one such example; indeed, Figure 7, used to argue for the presence of an arms race above, also shows abrupt retreats in generator complexity. To help us discover the adaptive utility of these sudden behavioral changes, Figure 11 plots, from top to bottom, average population fitness of the friendly predictors, hostile predictors, and generators, and order of the most fit generator over evolutionary time. We see that, on average, generators are able to behave more predictably to their friendly partners than to their hostile opponents almost throughout the run, as generator scores rarely fall below 0.5; average generator fitness increases, by definition, with increased difference between average friendly and hostile predictor fitness.

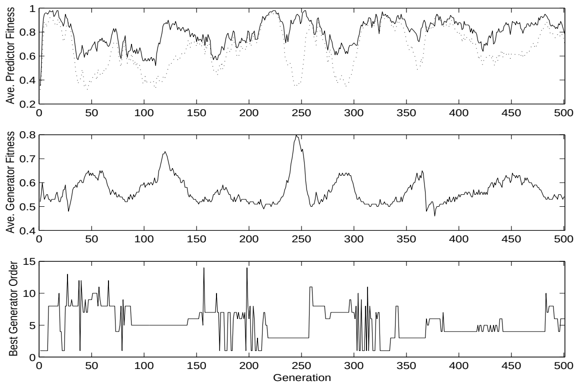

Figure 11: Fitness Averages and Best Generator Order.

We can develop an intuition of the dynamics in this run if we consider how generator order corresponds to fitness; particularly, we wish to pay attention to when, and how frequently, generator order changes. The periods of greatest stability in generator order always span the onset and duration of a period of generator success, that is, a period during which generators and their friendly predictor partners are most easily and effectively able to cooperate without falling prey to the hostile opponents. Generators stably maintain relatively low-order behavior during these periods.

Nevertheless, this very stability allows the hostile predictors to eventually learn the current behavioral convention, as evidenced by their fitness values. Once both populations of predictors are of comparable ability, generator fitness is minimal. Empirically, this tends to be the moment at which the order of champion generators becomes unstable, often alternating between very high-order and low-order behaviors. We conjecture that this period of generator instability injects noise into the evolutionary process of the predictor populations such that the two, presumably similar, populations once again diverge. When the two predictor populations become suitably differentiated, some medium-order generator is found that only the friendly predictors can predict. Thus, we enter a period of renewed generator stability and success. While further analysis is required to confirm this model of generator and predictor interaction, Figures 10 and 11 clearly show that rather than continuously improve predictor ability over the entire run, the system achieves mediocrity by repeatedly initiating short-lived arms races — a particularly interesting MSS.

We must recognize, however, that the desired arms race does not simply involve competition, but also enculturation towards convention. Presently, the generators and friendly predictors have no opportunity to develop their convention of cooperative behavior in isolation; once hostile predictors latch onto their current convention, the opportunity to evolve a more complex convention is long past. This issue may very well require a richer model of coevolution that encompasses both "public" and "private" interactions. Giving the generators the ability to distinguish friend from foe does not solve this problem, as a private sign of species "membership" must still be evolved.

The Question of Learnability

The central tenet of coevolution deems that an environment must be neither too difficult, nor too easy, for learning to take place. But, what precisely constitutes an extreme environment for a particular learner in a particular domain is a question usually left unasked out of faith that coevolutionary dynamics will correctly maintain a balanced environment, thus obviating the need to know. More seriously, most domains do not provide an obvious method of characterizing environment learnability, nor of constructing environments of arbitrary hardness. Our domain suggests ways in which to investigate not only the question of learnability, but also the question of what is really learned: to what extent might learning in one environment confer knowledge that is applicable in another? This kind of investigation is enabled only by the existence of a behavioral metric.

What's Too Easy? If a learner is not sufficiently challenged, nothing will be learned. The friendly predictors coevolved with the generators in the first half-game described above had such an impoverished environment. This half-game merely converged onto the simplest of conventions. When we play these predictors against the generators evolved in the full-game, they perform very poorly — 12% better than random, or roughly 56% prediction. Curiously, the generators frequently cause the predictors to behave very differently than they do against their "native" half-game opponents. Yet, the performance the predictors do achieve above random stems from games where they predict all *0*s and the generator has slightly more *0*s than *1*s.

What's Too Hard? If a learner is overwhelmed, nothing will be learned. Recall that, in the competitive half-game with scoring method B, the generators become reasonably complex and the hostile predictors perform only 6% better than random. The predictors do not appear to learn at all over the run. Consequently, we might assume that the environment provided by the generators is too hard. Yet, when we play these very same predictors against the generators from the full-game depicted in Figure 7, we are surprised to see that the predictors have, in fact, learned *something*. Figure 12a shows that some of these predictors actually perform as well as those coevolved in the full-game itself, though the skill displayed by the group of half-game predictors is very inconsistent.

This result begs the question of how well the best full-game predictors fare against the complex generators of the half-game. Indeed, they perform very poorly. This time we are not surprised, however, because we know from comparing Figures 5 and 7 that the full-game did not produce generators as complex as those seen in the competitive half-game. The full-game predictors cannot reasonably be expected to perform well versus generators much harder than those against which they evolved. Thus, the half-game generators confer adaptiveness to the full-game generators, but not vice versa.

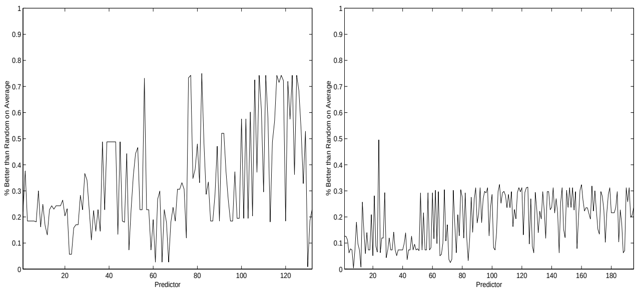

Figure 12: A (left): Performance of champion predictors of competitive half-game vs. champion generators of full-game; B (right): Predictors evolved against hand-built, chaotic generator show no improvement over evolutionary time against champion generators of first full-game.

Can we construct an environment that is too difficult for substantial learning to occur? To find out we use the logistic function as the sole sequence generator (instead of a population of neural networks) against which we evolve predictors. When the logistic function's behavior is at chaos and its output values are thresholded, it yields a binary sequence of true entropy $h = 1.0$. Since this generator will cause, on average, predictors to perform no better than random guessing, predictors will all receive the minimum fitness of zero with scoring method B. This makes the system behave equivalently to random search. Figure 12b shows that predictors thus evolved generally perform no better than 30% above random (65% prediction) when played against the full-game generators. Indeed, the lion's share of this 30% stems from predicting the single band of similar generators numbered 11 through 25 in Figure 8. Therefore, this pathological control is a significantly harder environment for evolving general prediction ability than the complex generators of the half-game. The prediction task is non-trivial and does not yield to simple random search.

Maintaining the Balance That cooperative half-game predictors are ill-prepared to play against full-game generators is not a particularly compelling result.

But, we also find that competitive half-game predictors perform inconsistently against full-game generators, and predictors evolved against the logistic function do poorly. Finally, predictors evolved from a different run of the full-game do relatively well against the full-game generators of Figure 7, ranging mostly from 30% to 60% above random prediction. We now see a picture consistent with the hypothesis of learnability — coevolutionary progress is heightened when a balance of skill can be maintained between participants.

Conclusions

The information-theoretic tools described in this paper allow us to measure the complexity of an evolved generator and construct a predictor that uses an arbitrarily large order statistic. Conspicuously, we do not directly build generators of known complexity, but rather we evolve them. Further, in this paper the power of evolved predictors is measured only indirectly, with respect to the evolved generators. We have recently built new tools that will allow us to directly measure predictor power and construct regular generators of arbitrary complexity.

The experiments described here are suggestive of the wide variety of questions our domain allows to be expressed. Our game provides not only a powerful metric of behavior, but also the ability to explore convergent and competitive dynamics and their interaction. The domain allows us to begin refining key notions in coevolutionary learning, namely arms-race dynamics, mediocre stable-states, and learnability.

By hand-building agents and environments we can artificially create situations that may arise during coevolution. This allows us to systematically test our game, our substrate, and coevolutionary dynamics. We discover, for example, that the ability of the substrate to successfully perform the opposing roles of our game does not guarantee that coevolution will find these solutions; just because a substrate can do X and Y does not mean that both will arise automatically when the substrate is placed in a coevolutionary framework.

We find that open-ended coevolution is not necessarily synonymous with a purely competitive framework; our game requires a mixture of cooperative and competitive pressures to avoid simple mediocre stable-states and closed-endedness. In this sense, our result agrees substantially with (Akiyama & Kaneko 1997).

Finally, because our domain allows us to characterize the difficulty of an environment, we can identify arms races that have broken because a participant has become too good. Indeed, we can begin to tease apart the many possible causes of system disfunction. While every domain has unique peculiarities, we believe the parsimony of our prediction game extends the validity and applicability of our results to other domains.

Acknowledgements

The authors gratefully acknowledge the many hours of conversation that have contributed to this work provided by Alan Blair, Marty Cohn, Paul Darwen, Pablo Funes, Greg Hornby, Ofer Melnik, Jason Noble, Elizabeth Sklar, and in particular Richard Watson. Thanks also to an anonymous reviewer for many helpful comments.

References

Akiyama, E., and Kaneko, K. 1997. Evolution of communication and strategies in an iterated three-person game. In Langton and Shimohara (1997), 150–158.

Angeline, P. J., and Pollack, J. B. 1994. Competitive environments evolve better solutions for complex tasks. In Forrest, S., ed., *Proceedings of the Fifth International Conference on Genetic Algorithms*, 264–270. Morgan Kaufmann.

Angeline, P. J.; Saunders, G. M.; and Pollack, J. B. 1994. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks* 5:54–65.

Axelrod, R. 1984. *The Evolution of Cooperation.* New York: Basic Books.

Cliff, D., and Miller, G. F. 1995. Tracking the red queen: Measurments of adaptive progress in co-evolutionary simulations. In Moran, F., et al., eds., *Third European Conference on Artificial Life*, 200–218. Berlin; New York: Springer Verlag.

Cliff, D., and Miller, G. F. 1996. Co-evolution of pursuit and evasion 2: Simulation methods and results. In Maes, P., et al., eds., *From Animals to Animats IV*, 506–515. MIT Press.

Hamming, R. W. 1980. *Coding and Information Theory.* Englewood Cliffs, NJ: Prentice-Hall, Inc.

Hillis, D. 1991. Co-evolving parasites improves simulated evolution as an optimization procedure. In Langton, C.; Taylor, C.; Farmer, J.; and Rasmussen, S., eds., *Artificial Life II (1990).* Addison-Wesley.

Isaacs, R. 1965. *Differential Games.* New York: John Wiley and Sons.

Kolen, J. F., and Pollack, J. B. 1994. The observer's paradox: Apparent computational complexity in physical systems. *The Journal of Experimental and Theoretical Artificial Intelligence* (Summer).

Langton, C. G., and Shimohara, K., eds. 1997. *Artificial Life V (1996).* MIT Press.

Pollack, J. B.; Blair, A.; and Land, M. 1997. Coevolution of a backgammon player. In Langton and Shimohara (1997).

Zhu, H., and Kinzel, W. 1997. Anti-predictable sequences: Harder to predict than a random sequence. *(Submitted).*