Towards Metrics and Visualizations Sensitive to Coevolutionary Failures

Ari Bader-Natal and **Jordan B. Pollack**
DEMO Lab, Computer Science Department
Brandeis University, MS018
Waltham, Massachusetts 02454–9110
{ari, pollack}@cs.brandeis.edu

Abstract

The task of monitoring success and failure in coevolution is inherently difficult, as domains need not have any external metric to measure performance. Past metrics and visualizations for coevolution have been limited to identification and measurement of *success but not failure.* We suggest circumventing this limitation by switching from "best-of-generation"-based techniques to "all-of-generation"-based techniques. Using "all-of-generation" data, we demonstrate one such techique – a *population-differential* technique – that allows us to profile and distinguish an assortment of coevolutionary successes and failures, including arms-race dynamics, disengagement, cycling, forgetting, and relativism.

Introduction

Coevolution requires no domain-specific notion of objective fitness, enabling coevolutionary algorithms to learn in domains for which no objective metric is known or for which known metrics are too expensive. But this benefit comes at the expense of accountability, as there is consequently no external metric with which to measure an algorithm's performance. Responses to this feedback void have come in the form of propositions for dynamics-based progress metrics. The most frequently used metrics have all been based upon Current Individual versus Ancestral Opponent (CIAO) plots (Cliff & Miller 1995). The CIAO plot offers useful feedback on performance, but does have limitations. In this paper, we focus primarily on one such limitation: the inability to provide feedback on coevolutionary failures. We present a related alternative, which could be called a Current Population versus Ancestral Opponent (CPAO) plot. We then offer one metric based upon the data in this plot.

One inherent difficulty in proposing metrics for processes lacking objective fitness valuations is that such metrics cannot be proved accurate. In order to address this, we examine a simple coevolutionary domain that is measurable, and look for corroborating results. The simple domains used for algorithmic auditing in this paper are the Numbers Games, introduced in (Watson & Pollack 2001). This is particularly relevant, as that work focused on addressing counter-productive behaviors in coevolutionary systems, often responsible for the failures for which we are interested in acquiring feedback. The final example presented uses a domain – Rock-Paper-Scissors – that possesses no objective metric. While the results drawn cannot be corroborated by such a metric, they are consistent with the cyclic nature of the game. We present this set of examples after reviewing existing CIAO-based techniques and presenting a CPAO-based alternative.

Best-of-Generation Techniques

Analysis based on generation tables was first proposed in coevolution by (Cliff & Miller 1995) based on CIAO data, and this work has subsequently been explored and built upon in several ways, including the Masters Tournament (Floreano & Nolfi 1997), the Dominance Tournament (Stanley & Miikkulainen 2002), and the Hall of Fame (Rosin & Belew 1997). In two-population coevolution, a *generation table* assigns the table rows to the first population's sequence of generations, and assigns table columns to successive generations of the second population. Internal table entries contain the results of evaluating the combination of the corresponding row and column generations. For data visualization, Cliff and Miller turn their tables into bitmap images (one pixel per table entry), and this paper employs a slightly modified version of that pixel-per-entry approach.[1]

This organization of data is valuable in making apparent the Red Queen effect: values drawn from evaluations along the table's diagonal[2] are simply incomparable to one another. Graphs displaying this *instantaneous fitness* over time are excellent illustrations of the Red Queen effect (see Fig. 1.) Generation table values are only comparable if either the candidate or the test is kept constant. For example, if one knows how a candidate at time t performs against some test T and one knows how the candidate at time $t+1$ performs against that same test, comparing the results may provide an indication of progress over time. If the second candidate were evaluated against something other than T, however, the comparison of results could no longer claim to be a valid in-

[1] The figures included in this work are oriented differently from Cliff and Miller, however. In this paper, the initial generation is placed in the upper-left. Additionally, the data for the entire generation table is calculated here.

[2] Specifically, the diagonal for which the i^{th} candidate generation is evaluated using the i^{th} test generation.

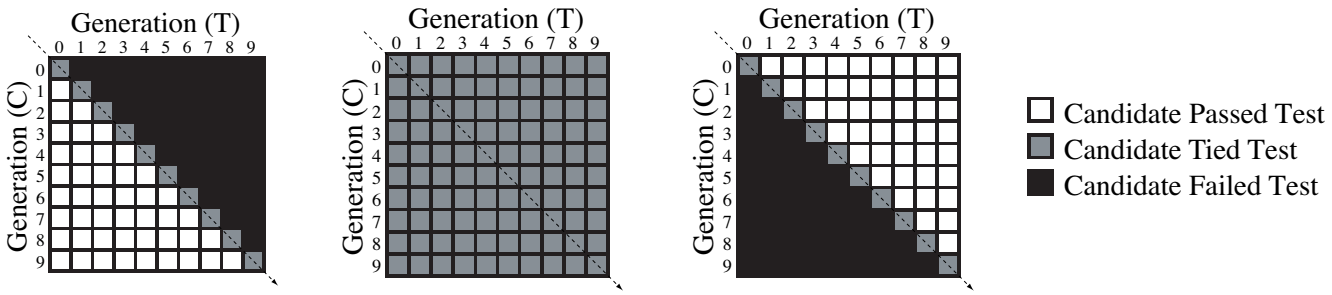

Figure 1: These three generation tables were constructed to demonstrate that dramatically different simulations can yield identical Instantaneous Fitness data. The instantaneous fitness of each simulation is based only on the 10 cells along the dotted diagonal line. Note that in all figures including generation tables in this paper, candidate generations increment from top to bottom, and test generations increment from left to right. Each cell in the table represents an evaluation of the corresponding row (candidate) and column (test).

dicator of progress.[3]

As noted in (Ficici & Pollack 2003), because of the history of single-objective fitness measurements, almost all approaches in the literature (including those cited above) concern themselves solely with the "best-of-generation" (BOG) member of each population. No other individuals are retained for analysis. This BOG approach appears particularly problematic for two reasons. First, results of an analysis can vary with the definition of "best" population member. (Cliff & Miller 1995; Floreano & Nolfi 1997; Stanley & Miikkulainen 2002) all adopt the Last Elite Opponent[4] criterion proposed in (Sims 1994), but it should be noted that any number of alternate definitions may be equally plausible. Indeed, the coevolutionary algorithm under examination may itself define "best" differently. Pareto coevolution, for example, would define "best" as the subset of individuals along the Pareto front. Second, while BOG-based analysis may give insight into algorithmic dynamics of the highly-successful individuals, it provides little about the population as a whole. While no claims were made about these monitors regarding sensitivity to failure, it ought to be noted that these BOG techniques are not sufficient to detect the common coevolutionary failures detailed in (Watson & Pollack 2001).

All-of-Generation Techniques

We introduce an alternative that addresses the shortcomings of BOG-based methods in monitoring failure based upon "all-of-generation" (AOG) data evaluation. For the sake of generality, the technique presented here is designed for simulations involving two asymmetric populations, but it is equally applicable to symmetric two-population simulations and to single-population simulations (via partitioning.) Adopting terminology from (Bucci & Pollack 2003b), we refer to these as populations of *candidates* and *tests*. This choice is meant to draw attention to the type of feedback resulting from evaluating a pair of elements. In order to minimize domain-specificity, the result of a *candidate-test* evaluation is simply one element of the ordered set $R = \{candidateFailedTest < candidateTiedTest < candidatePassedTest\}$, as in (Bucci & Pollack 2003b).

Population-Grained Evaluation

Where an entry in Cliff and Miller's generation table is the result of evaluating the "best" candidate against the "best" test for the specified generations, an entry in an AOG-based generation table must somehow represent the result of evaluating *all* candidates against *all* tests for the specified generations. Where individual-grained evaluation is well-defined, population-grained evaluation must be introduced. We simply average the numeric results of evaluating all candidates against all tests:[5]

$$PopEval(C_i, T_j) = \frac{\sum_{c \in C_i} \sum_{t \in T_j} eval(c,t)}{|C_i||T_j|}$$

where C_i is the i^{th}-generation candidate population and T_j is the j^{th}-generation test population. Numeric values result from implementing the ordered set R = $\{-1 < 0 < 1\}$. This results in scalar values between -1 (if all candidates fail all tests) to 1 (if all candidates pass all tests.) In the figures in this paper that include AOG data graphically, we map this scalar value to a grayscale value (where -1 is pure black, 0 is 50% gray, and 1 is pure white.)

Example: CIAO vs. CPAO

The difference between BOG and AOG data can now be elucidated with a simple example. Consider a two-population coevolutionary process that progresses for 10 generations, in

[3] Note that the Master Tournament (Floreano & Nolfi 1997) respects value comparability by restricting aggregations to within rows and or within columns. The Master Tournament attributes the summation of each row and column of a given generation as that generation's subjective fitness.

[4] An individual is defined to be the "best" of its generation if it outperforms its peers when pitted against the "best" member of the previous opponent generation.

[5] Averaging may not be appropriate for all applications. We use it here for simplicity.

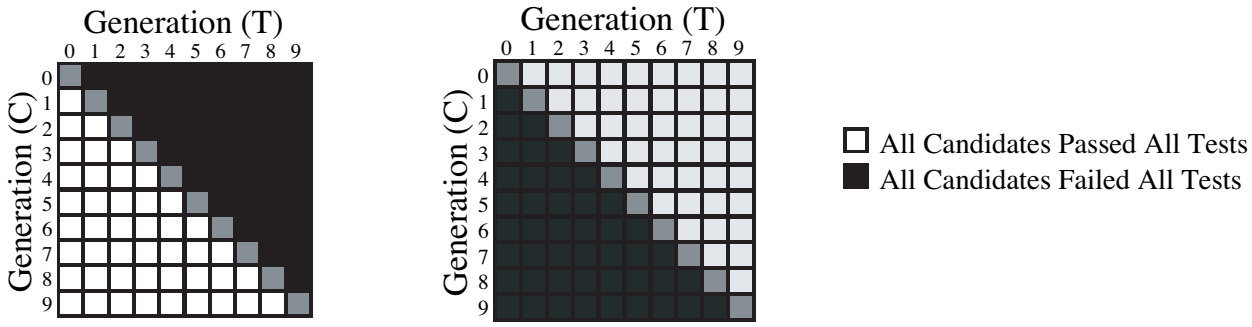

Figure 2: BOG data *(left)* and AOG data *(right)* can offer notably conflicting views of the same coevolutionary process. These generation tables were generated from a coevolutionary process described in the section entitled "Example: CIAO vs. CPAO". Each table cell is shaded based on the percentage of candidates (in the corresponding row) passing/failing the tests (in the corresponding column.)

which each population contains 10 individuals. In this simulation, let each individual be represented an integer, and let all individuals initially start at 0. Candidate/test evaluation is as follows: If a candidate is a larger integer than the test, the candidate "passes" that test. If a candidate is the same integer as the test, the candidate "ties" that test. If a candidate is a smaller integer than the test, the candidate "fails" that test. (Since all individuals are initialized at 0, all members of the candidate population initially "tie" all members of the test population.) Suppose the coevolutionary process proceeds as follows: Over the course of the 10 generations, one candidate and one test each mutate at a rate of +1 per generation, while the other nine individuals in each population mutate at a rate of -1 per generation. When the BOG and AOG data is plotted for this coevolutionary process, the pictures that emerge (Fig. 2) are dramatically different.

This simulation, drawn from one-dimensional variation of the Numbers Game domain (Watson & Pollack 2001), is provided as an example of how different the two sets of data may be. The BOG data suggests that the process exhibited and sustained some positive progression throughout the simulation, while the AOG data offers a different perspective, in which most of the change in the two populations was negative, and only a small fraction of the populations exhibited any positive progression. The value – and drawbacks – of both approaches are apparent here. While BOG data is not appropriate for all applications, neither is AOG data. It is up to the experimenter to select an approach that is appropriate to the task at hand. AOG-based techniques expand the set of available progress monitoring approaches, allowing for feedback that had previously been ignored or suppressed.

Computational Complexity

Clearly this change from BOG- to AOG-analysis increases the computational complexity of evaluation. Assuming population sizes remain constant (at $|C|$ candidates and $|T|$ tests per population), a g-generation simulation can be described as follows: a simple BOG analysis requires $g^2 + g|C| + g|T|$ evaluations (where the second and third terms are the cost of computing the Last Elite Opponent), while a simple AOG-based analysis requires $g^2|C||T|$ evaluations. While AOG-based analysis may not be feasible for domains in which evaluations are computationally intensive, much of the additional computational burden of switching from BOG to AOG can be alleviated by implementing a *memory policy*.

Memory Policies

Rather than computing every table entry in the generation table, one can restrict computation to a *representative subset* of entries, and base analysis on only this data. While we will not address the issue of optimally specifying "representative subset" here, we do provide examples of simple example subsets. Populating a generation table, as described above, involves evaluating all previous generations of one population against the current generation of the other population, and vice versa. This can be recast as a *memory-maintenance* operation: At each generation, add the new test population to a "test-memory", add the new candidate population to a "candidate-memory," and remove nothing from either memory. Then evaluate the new candidate population against everything in the test-memory and evaluate the new test population against everything in the candidate-memory.

Viewed as a "lossless" memory policy, the full generation table is defined simply: add each new generation and never remove any old generations. Two simple memory policies – both originally suggested in (Cliff & Miller 1995) – are presented below and illustrated in Fig. 3. The coevolutionary simulation examined in this figure is from a Pareto hill-climbing algorithm attempting the Compare-on-One variant of the Numbers Game, introduced in (de Jong & Pollack 2003). In this game, a hill-climber is used in which candidate selection is based on Pareto Dominance and test selection is based on *informativeness*, as defined in (Bucci & Pollack 2003b). The simulation was run for 400 generations with a fixed population size of 25 in both populations.[6]

[6]The mutation rate in all examples in this paper is 1.0. Numbers Game representation is integer-based, with mutation operating as follows: A pseudorandom, Gaussian distributed number (with mean 0.0 and standard deviation 1.0) is scaled by 4 (fixed mutation size), rounded to the nearest integer, and added to the current value.

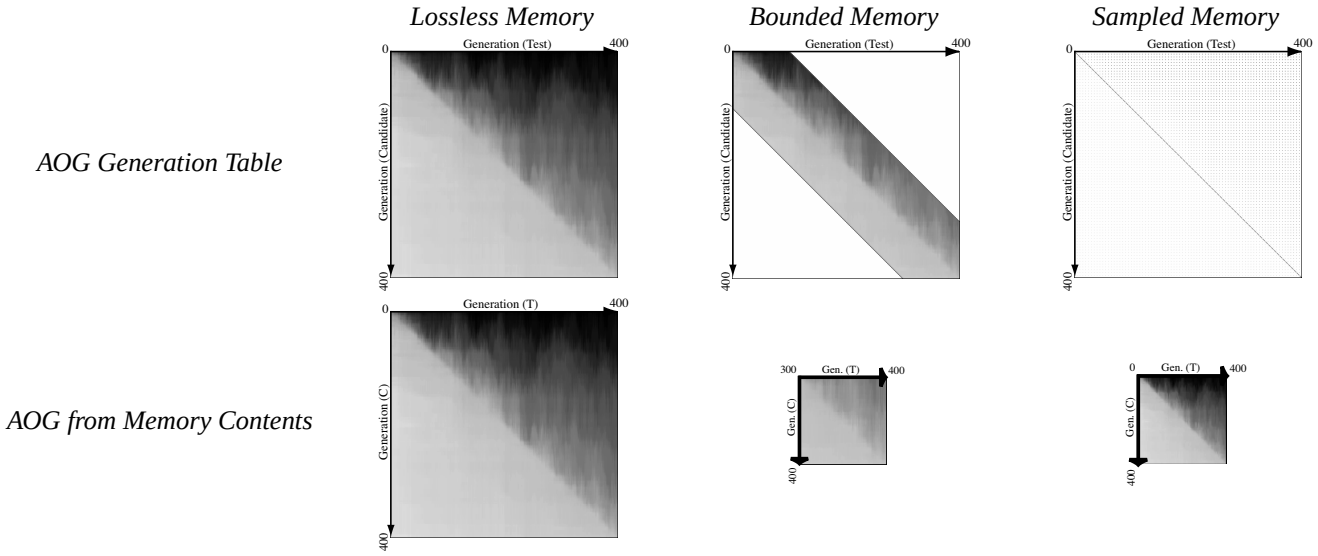

Figure 3: Illustration of three memory maintenance policy implementations. The upper images display all collected data in the generation table, and the lower images display evaluations based only on these memory contents following the *final generation* of the simulation. *Note on subsequent AOG figures: The grayscale percentage of each pixel is set to the percentage of all corresponding evaluations for which the candidate fails the test.*

Lossless Memory The AOG techniques described above can be considered "lossless". All generations in a simulation are added to the generation table and never removed, and are actively evaluated for every generation of the simulation. This is implemented as a list, to which each new generations is appended, in turn. This memory policy, as stated above, requires $g^2|C||T|$ evaluations for the analysis.

Sliding-Window Memory By implementing the memory as a fixed-size FIFO queue, the size of the memory can be easily bounded. A sliding-window memory, given window bound $b < g$, requires $(2bg - b^2)|C||T|$ evaluations. At one extreme, when the queue-size is set to the number of generations in the simulation, this memory is equivalent to the lossless memory policy described above. At the other extreme, a queue of size 1 effectively eliminates the notion of memory entirely, and the computation required is limited to that of the algorithm itself. In between lies a spectrum of tradeoff between accuracy and efficiency.

When the period of some cyclic behavior exceeds the memory window size, that cycle will not be discernible in the resulting AOG data. Similarly, other behaviors that exceed the memory window size may also yield misleading data. An example of one such behavior is illustrated in Fig. 4. A single coevolutionary process is plotted twice. In this process, a fitness-proportional coevolutionary algorithm is used on the Intransitive Numbers Game, with 15 individuals in each population and 600 generations computed. In the left-hand image, a 50-generation memory window is used, and in the right-hand image, a 300-generation window is used. The discrepancy between these two images suggest that the 50-generation memory offers an incomplete, misleading view of the process. The contents of this memory cannot be considered a representative subset of the memory.

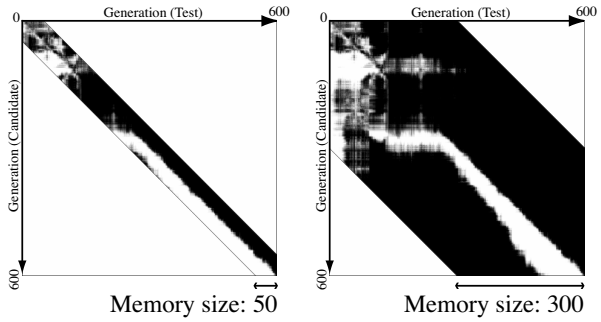

Figure 4: Increasing the sliding-window size b may reveal behavior that was not apparent with the shallower memory.

Sampled Memory Dramatic computational savings can be achieved by selectively adding populations to memory. If only every j^{th}generation is added to the memory, computation is reduced by a factor of j^2. For further savings, sampling can be used in conjunction with a sliding-window, requiring only $(2b(g/j) - b^2)|C||T|$ evaluations.

Concept-Specific Memory The population-differential monitor can simulate a Dominance Tournament (Stanley & Miikkulainen 2002) simply by implementing the following memory policy: If the best individual (according to the Last Elite Opponent) of the current generation beats all members of the other population in memory, add that individual to the memory. The individuals in memory are then the generated sequence of dominant strategies. This ability to easily simulate the Dominance Tournament may lead to better understanding the similarities and differences between these two approaches.

Population-Differential Analysis

Next we construct a perfomance measure based on the data available in the memory. Nolfi and Floreano addressed this by averaging data per generation (Floreano & Nolfi 1997), but this can make certain trends less apparent. The *cycling problem* associated with games with intransitivities, is an example of one such trend. As an alternative, we propose a comparison between the population in question and the oldest population in memory as a better indicator. The method presented below is appropriate for two-outcome evaluations (e.g. $R = \{loss < win\}$), and certain three-outcome evaluations (e.g. $R = \{loss < tie < win\}$). To begin, we define a *population comparator* (PC) conditionally:

$$PC_{T_k}(C_i, C_j) =$$

$$\begin{cases} 1, & \text{if } PopEval(C_i, T_k) > PopEval(C_j, T_k) \\ 0, & \text{if } PopEval(C_i, T_k) = PopEval(C_j, T_k) \\ -1, & \text{if } PopEval(C_i, T_k) < PopEval(C_j, T_k) \end{cases}$$

where $i > j$, C_i and C_j are the i^{th} and j^{th} candidate generations, respectively, and T_k is the k^{th} test population. Candidate-based comparators are defined similarly:

$$PC_{C_k}(T_i, T_j) =$$

$$\begin{cases} 1, & \text{if } PopEval(C_k, T_i) < PopEval(C_k, T_j) \\ 0, & \text{if } PopEval(C_k, T_i) = PopEval(C_k, T_j) \\ -1, & \text{if } PopEval(C_k, T_i) > PopEval(C_k, T_j) \end{cases}$$

where $i > j$, T_i and T_j are the i^{th} and j^{th} test generations,respectively, and C_k is the k^{th} candidate population. Intuitively, a population comparator reflects *directionality* of change over time.

The *candidate-population performance* at generation *i* is defined to be the average of the population comparators between the newest candidate population (C_i) and the oldest candidate population currently in the memory, with respect to all test populations currently in memory:

$$CandPerf_i = \frac{\sum_{T_k \in T} PC_{T_k}(C_i, C_{oldest})}{|T|}$$

Similarly, the *test-population performance* at generation *i* is defined to be the average of the *population comparators* between the current test population and the oldest test population in the memory, with respect to all candidate populations currently in memory:

$$TestPerf_i = \frac{\sum_{C_k \in C} PC_{C_k}(T_i, T_{oldest})}{|C|}$$

The population-differential metric, PD, is simply the average of these two performance levels:

$$PDPerf_i = \frac{CandPerf_i + TestPerf_i}{2}$$

The PD method generates one scalar value per generation for each population, ranging from -1 (when the eldest population outperforms the current population in every possible way with respect to the memory) to 1 (when the current population outperforms the eldest population in every possible way with respect to the memory).

By restricting attention to only the current state and the oldest recorded state, a population-differential analysis does not reward or penalize for localized variability. Additionally, this reduces the computational complexity of performance calculation. (For a memory of size n, only $2n$ comparisons need to be calculated.) These performance measures reflect the *directionality of change* over available memory.

Results

In order to demonstrate the value of the population-differential performance monitor, the first four examples presented use *Numbers Game* variants as a domain (Watson & Pollack 2001).In these games, tests and candidates are each a point on a two-dimensional integer grid. Mutation simply moves an individual to a nearby location. Evaluation of candidate-test pairs is done as follows: In the Compare-on-One game, individuals are compared according to the integer value of the dimension in which the test is larger. In the Intransitive game, individuals are compared according to the integer value of the dimension in which they are closer. These domains are used here because they are simple and they offer an acceptable external metric[7] that can be used to support or challenge the PD monitor. The fifth example uses the Rock-Paper-Scissors game as a domain, showing that the performance monitor can provide useful insight into games with no such external metric.

We examine five known coevolutionary behaviors here, which are labeled as follows in the figures below: *arms-race dynamics*, *lock-in failure*, *variations*, *disengagement*, and *cycling*. For each of the five, we describe what an idealized PD profile of each behavior ought to be, then we examine actual sample runs to see how they compare. For each sample, a PD chart is presented along with the corresponding Objective Fitness chart and AOG-based grayscale map. All three graphs share the same scale along the x-axis, allowing the reader to visually align the data from all three images for simple visual analysis.

Behavior Profiles

Interpretation of such performance graphs with respect to coevolutionary success and failure can be profiled as follows: An *arms-race dynamic* would yield a sustained PD value at or near 1, as both populations consistently do better later in time than they did earlier in time. *Lock-in failure* would yield a sustained value near -1, as the opposite is generally true, with the exception of localized progress. *Variation* is merely meant to describe a simulation for which the objective fitness graph switches direction several times. We would expect the performance monitor to register above zero on the inclines and below zero on the declines. *Disengagement* will yield a sustained value at zero once gradient is lost. *Cycling* will be visually apparent in the grayscale memory map. Note that in all cases, the PD monitor will

[7]The objective fitness of a population is defined to be the average sum of individuals' x- and y-coordinates.

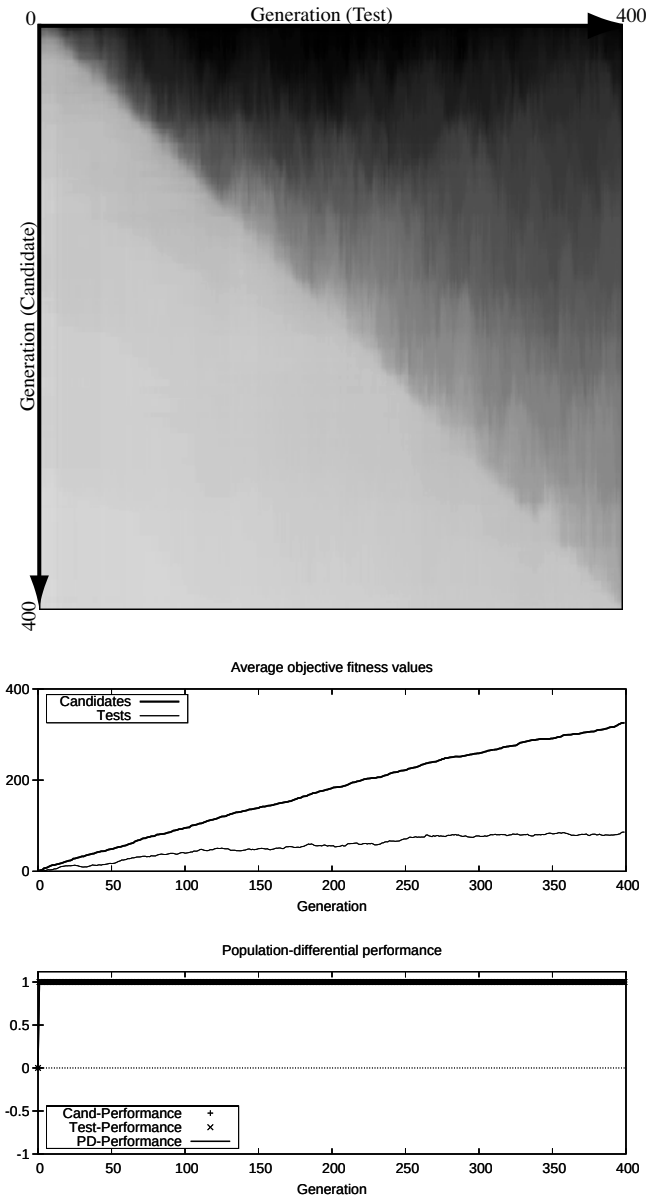

Figure 5: *Arms-race dynamics*. Pareto hill-climbing algorithm on Compare-on-One Numbers Game domain.

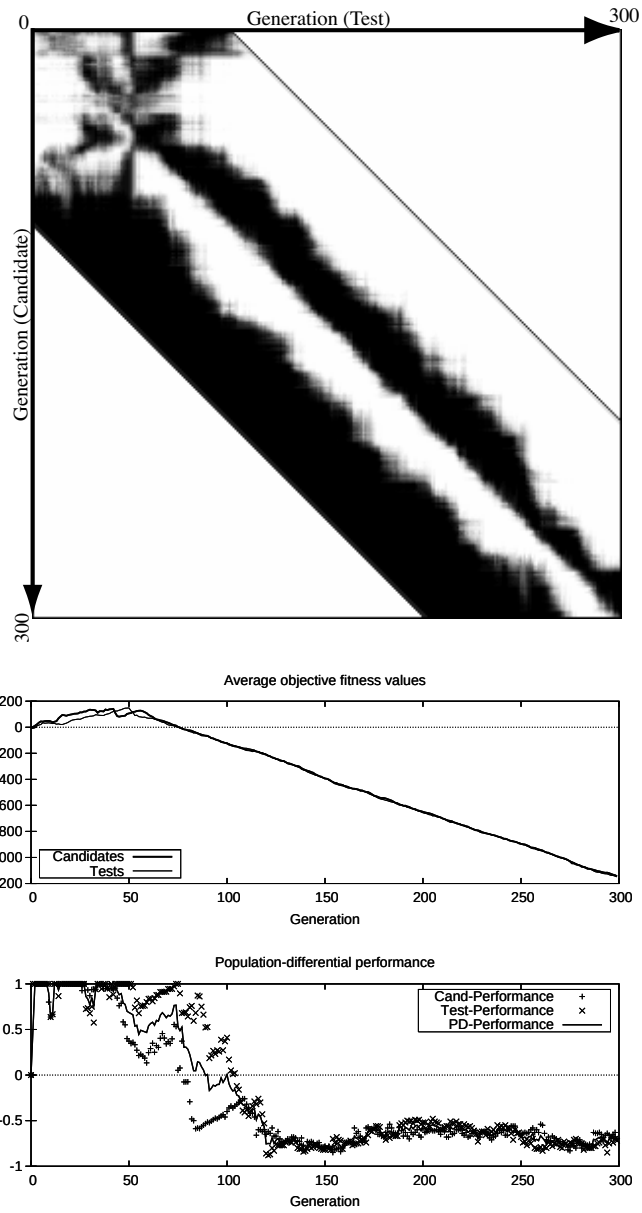

Figure 6: *Lock-in failure.* Fitness-proportional coevolutionary algorithm on Intransitive Numbers Game domain.

lag behind behaviors in the simulation by a fixed distance, determined by the size of the sliding memory-window.

Discussion

Each of the above theoretic performance profiles can now be compared to corresponding empirical samples.

For the *arms-race dynamic* example in Fig. 5, we include a 400-generation run (with fixed-sized populations of 25) from a Pareto hill-climbing algorithm (Bucci & Pollack 2003a) attempting the *Compare-on-One* Numbers Game (de Jong & Pollack 2003). Objective fitness in both populations steadily improves, and this is accurately characterized by the monitor. The AOG memory map reveals a remarkably smooth gradient. Note that progress in the two populations looks similar to the monitor, despite the varying rates revealed in the objective fitness graph. They appear equally good to the monitor because PD suggests the *direction* – and not *rate* – of change over time.

The *lock-in failure* of Fig. 6 was from a fitness-proportional coevolutionary algorithm attempting the *Intransitive* Numbers Game. The population sizes were fixed at 20 individuals, and the simulation ran for 300 generations. A short run of initial progress is recognized by the monitor, which is gradually replaced by a near-bottom value as the

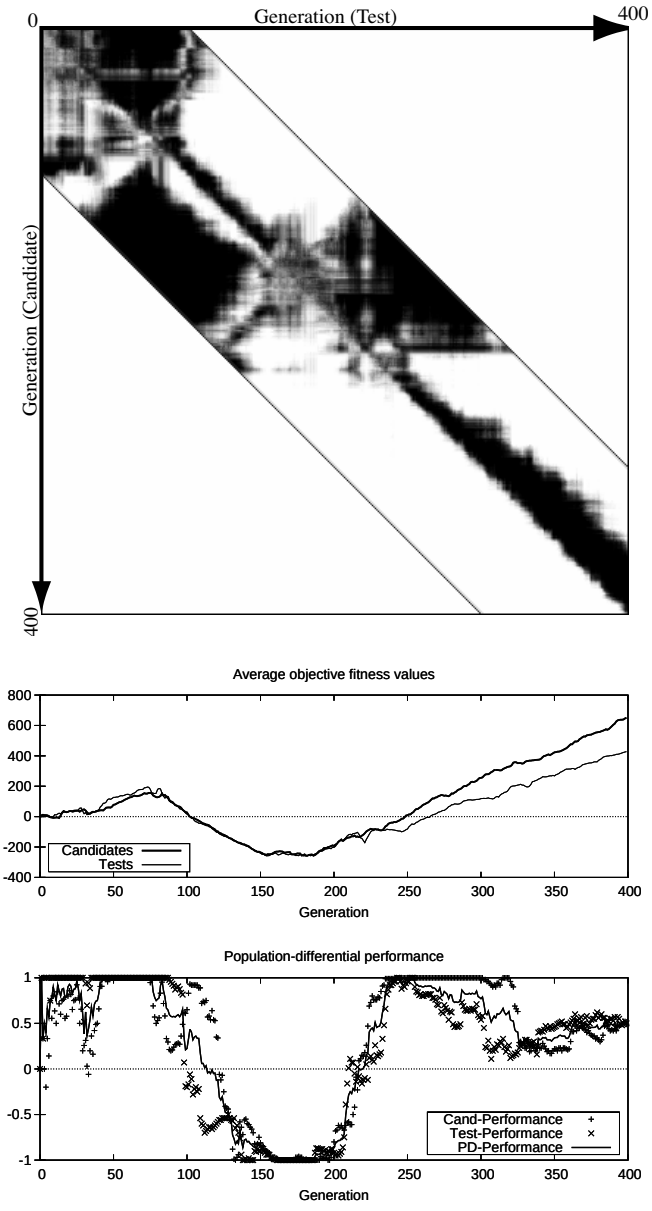

Figure 7: *Variation.* Fitness-proportional coevolutionary algorithm on Intransitive Numbers Game domain.

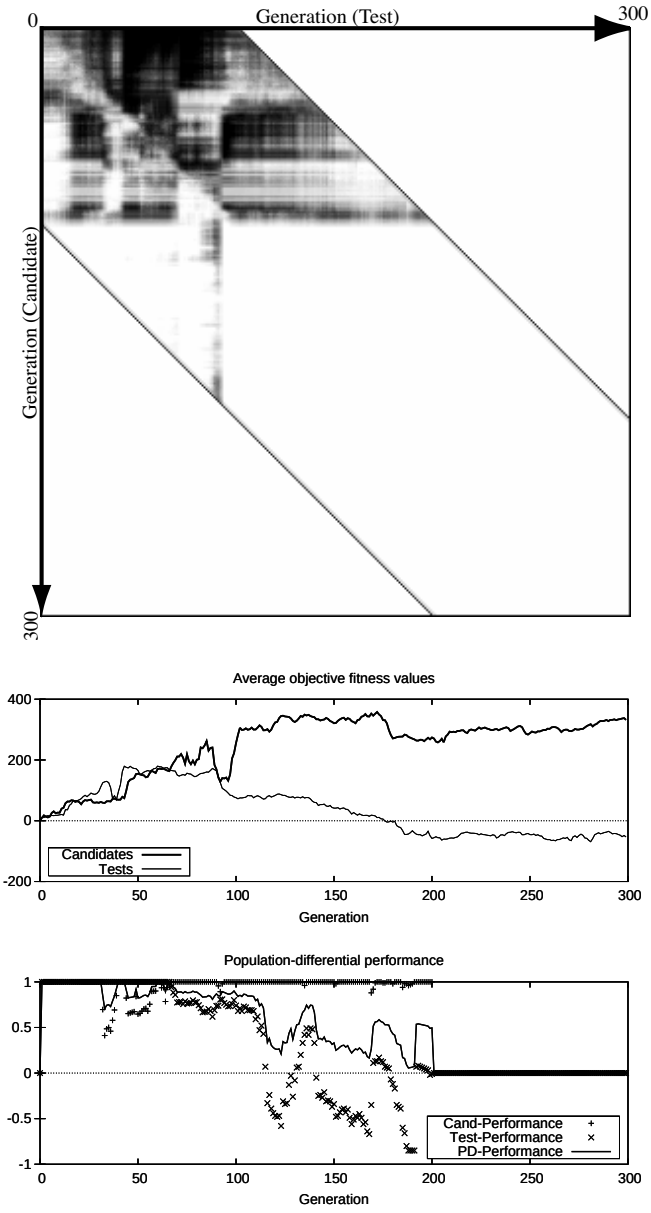

Figure 8: *Disengagement.* Fitness-proportional coevolutionary algorithm on Intransitive Numbers Game domain.

two populations settle in to a locally-improving but globally-detrimental pattern. Note how this pattern visually resounds in the grayscale map. Also, note how the 100-generation bounded-memory size causes a 100-generation lag between activity in the simulation (visible in the objective fitness) and the values in the performance monitor.

The *variation* evident in Fig. 7 was generated by a fitness-proportional coevolutionary algorithm again attempting the Intransitive Numbers Game. The population sizes were fixed at 20 individuals, and the simulation ran for 400 generations. Note that the mid-level value (settling between 0 and 1) of the monitor at the conclusion suggests that the final upswing was characterized by *learning-and-forgetting*, rather than a true *arms race*. The memory map image supports this, showing that only one of the two populations could retain its behavior over time.

The *disengagement* that occurs in Fig. 8 was generated by a fitness-proportional coevolutionary algorithm again attempting the Intransitive Numbers Game. The population sizes were fixed at 20 individuals, and the simulation ran for 300 generations. The disengagement is recognized once the memory-window slides beyond it, leaving a continuous trail at zero, with no progress beyond that. The notion of *loss-of-gradient* is visually apparent in the grayscale AOG data.

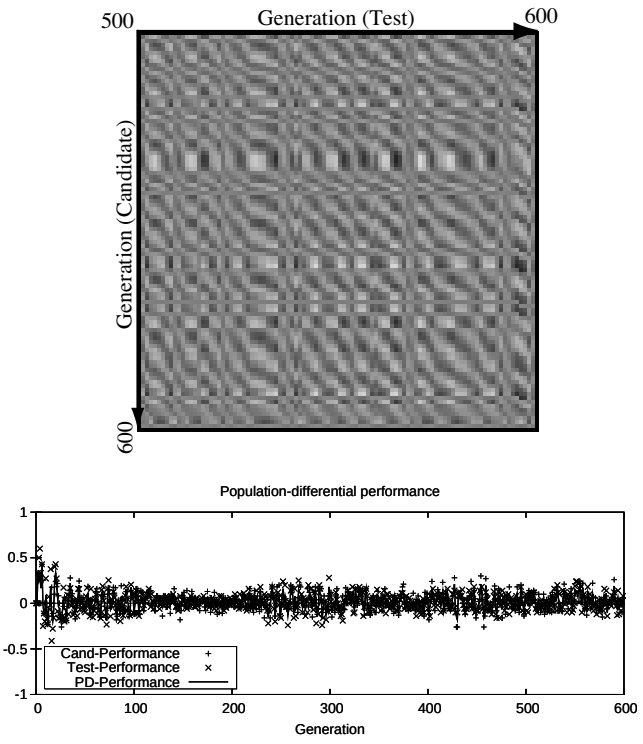

Figure 9: *Cycling.* Coevolutionary hill-climbing algorithm on Rock-Paper-Scissors domain. Note that no objective fitness graph is present, because no such fitness values are available for the Rock-Paper-Scissors domain. An enlarged 100-generation subsection of the AOG generation table is shown at top.

Finally, the *cycling* visible in Fig. 9 results from a coevolutionary hill-climbing algorithm attempting the game of Rock-Paper-Scissors. The population sizes were fixed at 20 individuals, and the simulation ran for 600 generations.[8] The *intransitive superiorities* inherent in this game lead the algorithm in circles (a Pareto hill-climber, for contrast, does not fall into such cycles.) The PD monitor stays near zero, and any sort of smoothing would make this even more apparent. Note that since this domain has no suitable objective fitness metric, no such graph can be included here. A subsection of the AOG-memory map is enlarged to display the visual appearance of cycles in memory.

Conclusion

The AOG-based framework introduced here seems particularly well-suited to the coevolutionary monitoring task, as it is sensitive to the full spectrum of anomalies detailed in (Watson & Pollack 2001). The major drawback of switching from a BOG-based technique to an AOG-based technique is the additional computational burden, but this can be alleviated through design of the memory-maintenance policy. The population-differential monitor then builds on this AOG-based data, just as the Masters and Dominance Tournaments build on BOG-based data. The richness of the results is significant, expanding the set of coevolutionary behaviors that can be effectively recorded and visualized. The ability to identify and track arms-races, Red Queen effects, and various anomalies discussed in this paper may lead to an ability to adaptively control coevolution while maintaining continuous learning.

[8] Individuals in the game are represented as either a Rock, Paper, or Scissors. Mutation randomly yields any one of these three states.

References

Bucci, A., and Pollack, J. B. 2003a. Focusing versus intransitivity: Geometrical aspects of coevolution. In Cantú-Paz, E., et al., eds., *Genetic and Evolutionary Computation - GECCO 2003*, volume 2723 of *Lecture Notes in Computer Science*, 250–261. Springer.

Bucci, A., and Pollack, J. B. 2003b. A mathematical framework for the study of coevolution. In De Jong, K. A.; Poli, R.; and Rowe, J. E., eds., *Foundations of Genetic Algorithms 7*. San Francisco: Morgan Kaufmann. 221–235.

Cliff, D., and Miller, G. F. 1995. Tracking the red queen : Measurements of adaptive progress in co-evolutionary simulations. In Moran, F.; Moreno, A.; Merelo, J.; and Chacon, P., eds., *Advances in Artificial Life: Proceedings of the Third European Conference on Artificial Life*, volume 929, 200–218. Berlin: Springer-Verlag.

de Jong, E. D., and Pollack, J. B. 2003. Learning the ideal evaluation function. In Cantú-Paz, E., et al., eds., *Genetic and Evolutionary Computation – GECCO-2003*, volume 2723 of *LNCS*, 274–285. Chicago: Springer-Verlag.

Ficici, S. G., and Pollack, J. B. 2003. A game-theoretic memory mechanism for coevolution. In Cantú-Paz, E., et al., eds., *Genetic and Evolutionary Computation – GECCO-2003*, volume 2723 of *LNCS*, 286–297. Chicago: Springer-Verlag.

Floreano, D., and Nolfi, S. 1997. God save the red queen! competition in co-evolutionary robotics. In Koza, J. R., et al., eds., *Genetic Programming 1997: Proceedings of the Second Annual Conference*, 398–406. Stanford University, CA, USA: Morgan Kaufmann.

Rosin, C. D., and Belew, R. K. 1997. New methods for competitive coevolution. *Evolutionary Computation* 5(1):1–29.

Sims, K. 1994. Evolving 3d morphology and behavior by competition. In Brooks, R. A., and Maes, P., eds., *Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems* $ArtificialLifeIV$, 28–39. Cambridge, MA, USA: MIT Press.

Stanley, K. O., and Miikkulainen, R. 2002. The dominance tournament method of monitoring progress in coevolution. In Barry, A. M., ed., *GECCO 2002: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, 242–248. New York: AAAI.

Watson, R. A., and Pollack, J. B. 2001. Coevolutionary dynamics in a minimal substrate. In Spector, L., et al., eds., *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*. Morgan Kaufmann.