A Population-Differential Method of Monitoring Success and Failure in Coevolution*

Ari Bader-Natal and Jordan B. Pollack

{ari,pollack}@cs.brandeis.edu
DEMO Lab. Brandeis University. Waltham, MA 02454

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 work on monitoring "progress" all strive to identify and measure success, but none attempt to identify failure. We suggest that this limitation is due to the reliance on a "best-of-generation" (BOG) memory mechanism, and propose an alternate "all-of-generation" (AOG) mechanism free of this limitation. Using AOG data, we propose a *population-differential* method for monitoring coevolution in arbitrary domains. With this method, we demonstrate the ability to profile and distinguish an assortment of coevolutionary successes and failures, including arms-race dynamics, disengagement, cycling, forgetting, and relativism.

1 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. Several counter-productive behaviors have been identified in the literature using simple but measurable domains such as the *numbers game*, introduced by Watson and Pollack in [10]. While such domains offer opportunities for *algorithmic auditing*, similar feedback cannot be obtained from an arbitrary coevolutionary domain.

After reviewing existing BOG monitoring techniques (including Cliff and Miller's CIAO data [3], Nolfi and Floreano's Masters Tournament [6], and Stanley and Miikkulainen's Dominance Tournament [9]), we present an AOG alternative. Upon this framework we develop a method called *population-differential analysis*, and demonstrate that it can provide rich feedback about coevolutionary successes and failures. This feedback can be used both to refine variations in algorithms and to automate ways of controlling parameters for coevolution in novel domains.

* Brandeis CS Tech Report CS-04-252

2 Related Work: Best-of-Generation Techniques

Analysis based on *generation tables* was first introduced in coevolution by Cliff and Miller, and has been pursued further by others, including Floreano and Nolfi, and Stanley and Mikkulainen [6,9,3].[1] In dual-population coevolution,[2] 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 we follow their pixel-per-entry example with grayscale-maps.

This organization of data is valuable in making apparent the Red Queen effect: values drawn from evaluations along the table's diagonal[3] 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 indicator of progress.[4]

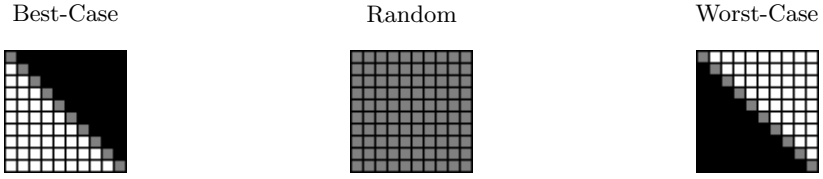


Fig. 1. An interpretation of the Red Queen effect with respect to generational tables. Three examples of dramatically different performance that correspond to the same *instantaneous fitness* graph. (with data taken from the diagonal.) In these images, as in all subsequent generation table images, candidate generations progress from top to bottom, and test generations progress from left to right. Each enlarged "pixel" represents some evaluation of the two corresponding row and column generations.

[1] Rosin and Belew later incorporated information from generation tables into the selection mechanism, in their *Hall of Fame* concept.[7]

[2] Specifically, one population is considered *candidates* and the other population is considered *tests*, following from Bucci and Pollack in [2].

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

[4] Note that the Master Tournament [6] 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.

As Ficici and Pollack note in [5], 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. [3,6,9] all adopt the *Last Elite Opponent*[5] criterion proposed by Sims in [8], 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.[6] 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 by Watson and Pollack in [10].

3 All-of-Generation Technique

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 [2], we refer to these as populations of *candidates* and *tests*. This choice, over common alternatives such as "learners and teachers" or of simply "population A and population B," 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* pairing is simply one element of the ordered set R = {*candidate-failed-test* ¡ *candidate-tied-test* ¡ *candidate-passed-test*}, as in [2].

3.1 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. A linear combination is implemented here. We simply average the numeric results of evaluating all candidates against all tests:

[5] 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.

[6] Pareto coevolution, for example, would define "best" as the subset of individuals along the Pareto front.

Definition 1. $PopEval(C_i, T_j) = \frac{\sum_{c \in C_i} \sum_{t \in T_j} eval(c,t)}{|C_i||T_j|}$

where C_i is the generation 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 -1.0 (if all candidates fail all tests) to 1.0 (if all candidates pass all tests.)

3.2 Computational Complexity of AOG Analysis

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[7] (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. Fortunately, much of the additional computational burden of switching from BOG to AOG analysis can be alleviated by implementing a *memory-maintenance policy*.

3.3 Reducing Complexity with 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[8] are presented below, and are illustrated in Fig. 2.[9]

[7] Recall that one evaluation takes one candidate and one test, and returns one element from set R = {*candidate-failed-test ¡ candidate-tied-test ¡ candidate-passed-test*}.

[8] These alternative policies were suggested in [3].

[9] The memory mechanisms described here are simply tasked with determining when to add and remove generations from accessibility. There is no goal here to collect the "best" elements, as in the memory mechanisms detailed in [5].

Lossless Memory The AOG techniques described above are "lossless" by default. All generations in a simulation are added to the generation table, and they 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 in Sec. 3.2, 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. 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. Fig. 3 illustrates how an overly-restrictive memory size could severely impair the value of the analysis. A sliding-window memory, given window bound $b < g$, requires $(2bg - b^2)|C||T|$ evaluations.

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, to $(g/j)^2|C||T|$ evaluations. For further savings, sampling can be used in conjunction with the sliding-window described above, requiring only $(2b(g/j) - b^2)|C||T|$ evaluations.

Concept-Specific Memory The population-differential monitor can simulate a Dominance Tournament simply by implementing a memory policy as follows: If the best individual (with respect to the *Last Elite Opponent*) of the current generation beats all members of the other population currently in memory, add that individual to the memory. At the end of the simulation, the individuals in memory are the sequence of dominant strategies that were generated.[10]

4 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 [6], but this makes most 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 *population comparator* (PC) is defined conditionally:

[10] While it is not generally interesting that concept-specific memory policies can be developed that coerce the population-differential monitor to perform other tasks, the ability to easily simulate the *Dominance Tournament* allows for better understanding similarities and differences between the two approaches.

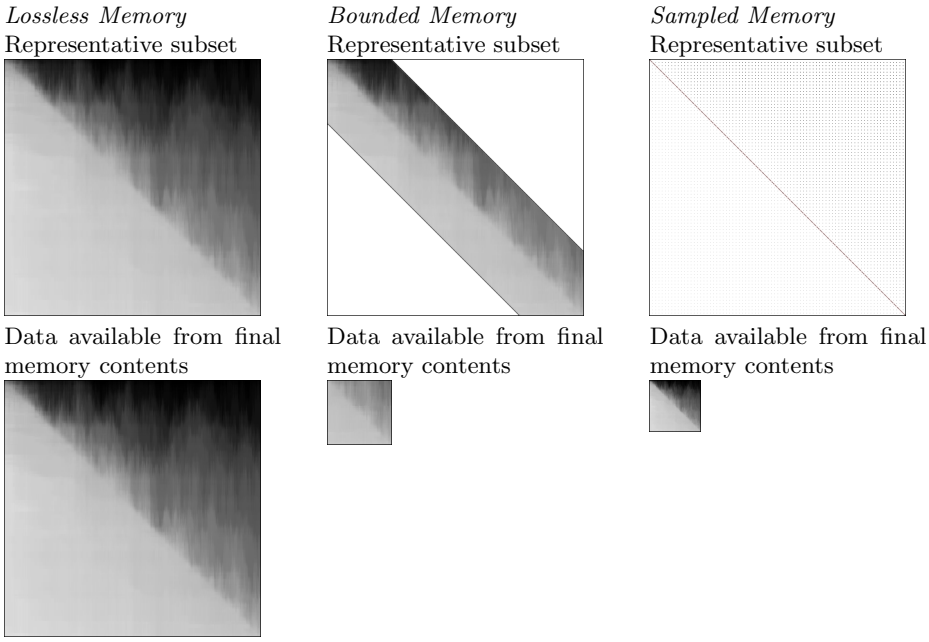


Fig. 2. Illustration of three memory maintenance policy implementations. The top images display the policy-specific subset of the generation table, and the bottom image displays evaluations based only on the memory contents at the final generation of the simulation. This simulation is from a Pareto hill-climbing algorithm attempting the *Compare-on-One* variant of the *numbers game*, introduced in [4]. (A hill-climber in which candidate selection is based on *Pareto dominance* and test selection is based on *informativeness*, as defined by Bucci and Pollack in [2].)

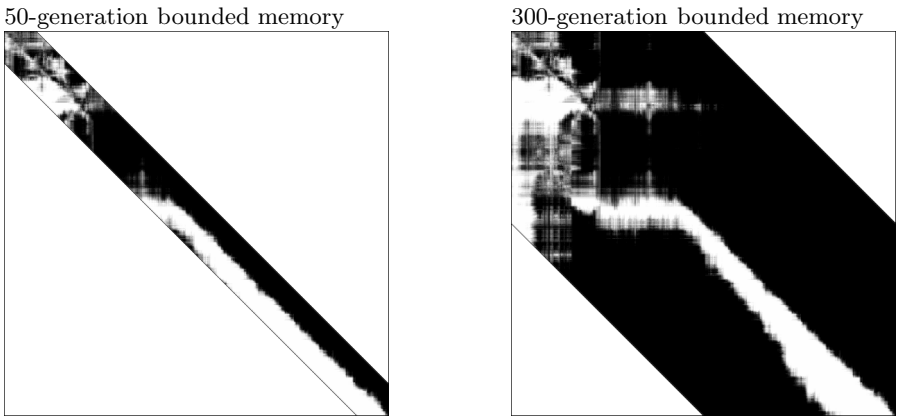


Fig. 3. Increasing the sliding-window size b may reveal behavior that was not apparent with the shallower memory. These images were drawn from a fitness-proportional coevolutionary algorithm attempting Watson and Pollack's *intransitive numbers game*. The 300-generation memory image shows that the 50-generation memory is not sufficient to derive a representative subset of AOG data for this simulation.

Definition 2. $PC_{T_k}(C_i, C_j) = \begin{cases} 1, \text{if } PopEval(C_i, T_k) > PopEval(C_j, T_k) \\ 0, \text{if } PopEval(C_i, T_k) = PopEval(C_j, T_k) \\ -1, \text{if } PopEval(C_i, T_k) < PopEval(C_j, T_k) \end{cases}$

where $i > j$, C_i and C_j are candidate generations, and T_k is a test population. Candidate-based comparators are defined similarly:

Definition 3. $PC_{C_k}(T_i, T_j) = \begin{cases} 1, \text{if } PopEval(C_k, T_i) < PopEval(C_k, T_j) \\ 0, \text{if } PopEval(C_k, T_i) = PopEval(C_k, T_j) \\ -1, \text{if } PopEval(C_k, T_i) > PopEval(C_k, T_j) \end{cases}$

where $i > j$, T_i and T_j are test generations, and C_k is a 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 (i^{th}) candidate population and the oldest candidate population currently in the memory, with respect to all test populations currently in memory:

Definition 4. $CandPerformance_i = \frac{\sum_{T_k \in T} PC_{T_k}(C_i, C_{oldest})}{|T|}$

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:

Definition 5. $TestPerformance_i = \frac{\sum_{C_k \in C} PC_{C_k}(T_i, T_{oldest})}{|C|}$

The *PC-Performance* is simply the average of these two performance levels:

Definition 6. $PCPerformance_i = \frac{CandPerformance_i + TestPerformance_i}{2}$

The PC-Perfomance method generates one scalar value per generation for each population, ranging from -1.0 (when the eldest population outperforms the current population in every possible way with respect to the memory) to 1.0 (when the current population outperforms the eldest population in every possible way with respect to the memory).

By restricting interest to the far reaches of the memory, a population-differential analysis does not reward or penalize for localized variability. The only change that is of interest is that between the population's ancestry and its current state. Additionally, this keeps the computational complexity of calculating performance low. (For a memory of size n, only $2n$ comparisons need to be calculated.) Intuitively, these performance measures reflect the *directionality* of change over available memory.

5 Results

In order to demonstrate the value of the population-differential performance monitor, the first four experiments presented use *numbers game* variants as a domain.[11] These domains are trivially simple and, more importantly, offer an acceptable external metric[12] that can be used to support or challenge the PC-Performance monitor. The fifth experiment 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 PC-Performance profile of each behavior ought to be, then we examine actual sample runs to see how they compare. For each sample, a PC-Performance 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.

5.1 PC-Performance 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 PC-Performance 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 PC-Performance monitor will lag behind behaviors in the simulation by a fixed distance, determined by the size of the sliding memory-window.

5.2 Discussion of Results

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

For the *arms-race dynamic* example in Fig. 4, we include a run from a Pareto hill-climbing algorithm [1] attempting the *compare-on-one* numbers game [4].

[11] In these games, tests and candidates are each a point on a two-dimensional grid. Mutation simply moves an individual to a nearby location. Evaluation of candidate-test pairs varies by game variant, and details are included in [10].

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

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 PC-Performance suggests the *direction* – and not *rate* – of change over time.

The *lock-in failure* of Fig. 5 was from a fitness-proportional coevolutionary algorithm attempting the *intransitive* numbers game. A short run of initial progress is recognized by the monitor, which is gradually replaced by a near-bottom value as the 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 graph) and the values in the performance monitor.

The *variation* evident in Fig. 6 was generated by a fitness-proportional coevolutionary algorithm again attempting the intransitive numbers game. 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.

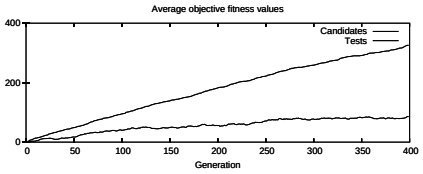
The *disengagement* that occurs in Fig. 7 was generated by a fitness-proportional coevolutionary algorithm again attempting the intransitive numbers game. The disengagement is recognized once the memory-window slides beyond it, leaving a continuous trail at zero progress beyond that. The memory map is particularly notable here, as the notion of *loss-of-gradient* is so visually apparent.

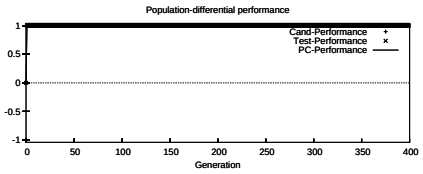
Finally, the *cycling* visible in Fig. 8 results from a coevolutionary hill-climbing algorithm attempting the game of Rock-Paper-Scissors. The *intransitive superiorities* inherent in this game lead the algorithm in circles (a Pareto hill-climber, for contrast, does not fall into such cycles.) The PC-Performance 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 sub-section of the AOG-memory map is enlarged to display the visual appearance of cycles in memory.

6 Conclusion

The AOG-based framework introduced here seems particularly well-suited to the coevolutionary monitoring task, as it is sensitive to the entire spectrum of known anomalies detailed by Watson and Pollack in [10]. 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 Master and Dominance Tournaments build on BOG-based data. The richness of the results is significant, as illustrated in the figures. With an ability to detect these behaviors, we will be able to

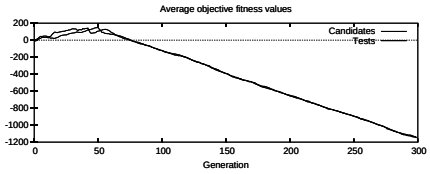
Fig. 4. *Arms-race dynamics.* Pareto hill-climbing algorithm on compare-on-one numbers game domain.

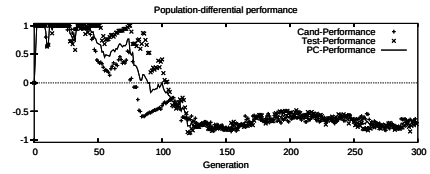


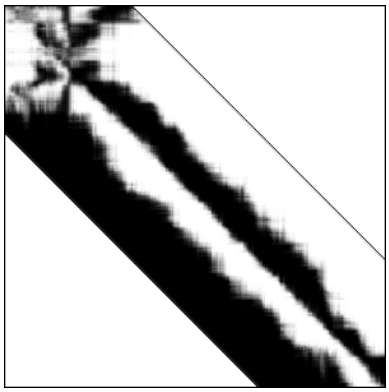


AOG data

Fig. 5. *Lock-in failure.* Fitness-proportional coevolutionary algorithm on intransitive numbers game domain.

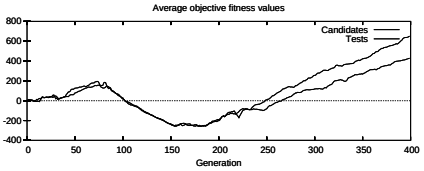


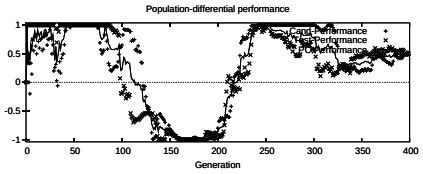


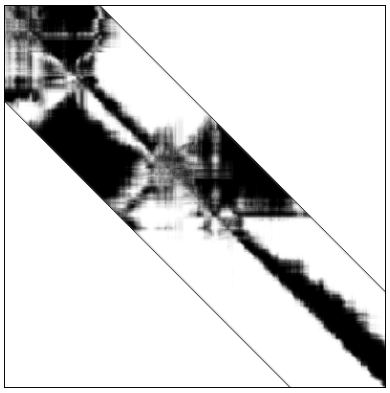


AOG data

Fig. 6. *Variation.* Fitness-proportional coevolutionary algorithm on intransitive numbers game domain.

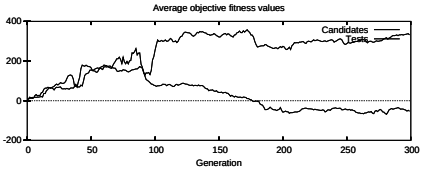


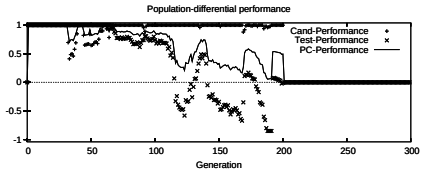


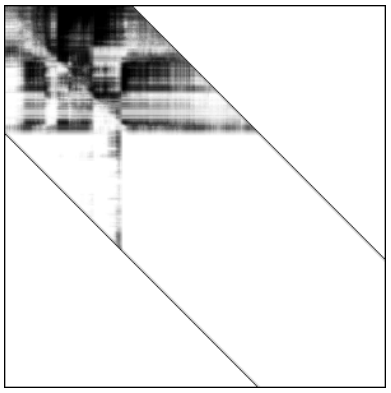


AOG-data

Fig. 7. *Disengagement.* Fitness-proportional coevolutionary algorithm on intransitive numbers game domain.

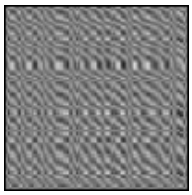




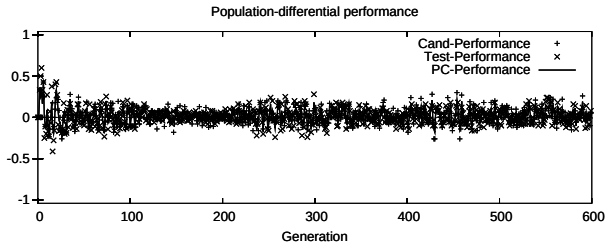


AOG data

Fig. 8. *Cycling.* Coevolutionary hill-climbing algorithm on Rock-Paper-Scissors domain.



AOG-data



apply coevolution to more complex unknown tasks, refining parameters while monitoring PC-performance. The ability to identify and track arms races, Red Queen effects, and disengagements may lead to an ability to adaptively control coevolution while maintaining continuous learning.

References

1. Anthony Bucci and Jordan B. Pollack. Focusing versus intransitivity: Geometrical aspects of coevolution. In E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller, editors, *Genetic and Evolutionary Computation - GECCO 2003*, volume 2723 of *Lecture Notes in Computer Science*, pages 250–261. Springer, 2003.
2. Anthony Bucci and Jordan B. Pollack. A mathematical framework for the study of coevolution. In Kenneth A. De Jong, Riccardo Poli, and Jonathan E. Rowe, editors, *Foundations of Genetic Algorithms 7*, pages 221–235. Morgan Kaufmann, San Francisco, 2003.
3. D. Cliff and G. F. Miller. Tracking the red queen : Measurements of adaptive progress in co-evolutionary simulations. In F. Moran, A. Moreno, J.J. Merelo, and P. Chacon, editors, *Advances in Artificial Life: Proceedings of the Third European Conference on Artificial Life*, volume 929, pages 200–218, Berlin, 1995. Springer-Verlag.
4. Edwin D. de Jong and Jordan B. Pollack. Learning the ideal evaluation function. In E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller, editors, *Genetic and Evolutionary Computation – GECCO-2003*, volume 2723 of *LNCS*, pages 274–285, Chicago, 12-16 2003. Springer-Verlag.
5. Sevan G. Ficici and Jordan B. Pollack. A game-theoretic memory mechanism for coevolution. In E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller, editors, *Genetic and Evolutionary Computation – GECCO-2003*, volume 2723 of *LNCS*, pages 286–297, Chicago, 12-16 2003. Springer-Verlag.
6. Dario Floreano and Stefano Nolfi. God save the red queen! competition in co-evolutionary robotics. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 398–406, Stanford University, CA, USA, 13-16 1997. Morgan Kaufmann.
7. Christopher D. Rosin and Richard K. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.
8. Karl Sims. Evolving 3d morphology and behavior by competition. In Rodney A. Brooks and Pattie Maes, editors, *Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems ArtificialLifeIV*, pages 28–39, Cambridge, MA, USA, 7 1994. MIT Press.
9. Kenneth O. Stanley and Risto Miikkulainen. The dominance tournament method of monitoring progress in coevolution. In Alwyn M. Barry, editor, *GECCO 2002: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, pages 242–248, New York, 8 2002. AAAI.

10. R. A. Watson and J. B. Pollack. Coevolutionary dynamics in a minimal substrate. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, , and Edmund Burke, editors, *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, 2001.