The MaxSolve Algorithm for Coevolution

Edwin de Jong
Institute of Information and Computing Sciences
Utrecht University
PO Box 80.089
3508 TB Utrecht, The Netherlands
dejong@cs.uu.nl

ABSTRACT

Coevolution can be used to adaptively choose the tests used for evaluating candidate solutions. A long-standing question is how this dynamic setup may be organized to yield reliable search methods. Reliability can only be considered in connection with a particular solution concept specifying what constitutes a solution. Recently, monotonic coevolution algorithms have been proposed for several solution concepts. Here, we introduce a new algorithm that guarantees monotonicity for the solution concept of maximizing the expected utility of a candidate solution. The method, called MaxSolve, is compared to the IPCA algorithm and found to perform more efficiently for a range of parameter values on an abstract test problem.

Categories and Subject Descriptors

F.0 [**General**]

General Terms

Algorithms, Reliability, Experimentation, Performance

Keywords

Coevolution, archive, reliability, reliable coevolution

Introduction

Coevolution (Barricelli, 1962, 1963; Axelrod, 1987; Miller, 1989, 1996; Hillis, 1990; Koza, 1992; Lindgren, 1992; Kauffman & Johnsen, 1992) offers methods for problems where the quality of individuals is determined by tests. For example, in two-player games, the quality of a first-player strategy is somehow determined by its outcome against all possible opponents. Other test-based problem domains include concept learning and function approximation.

Since the number of tests in a test-based problem can be very large, evaluating individuals on all possible tests is typically infeasible. Another option is to define a heuristic evaluation function, or to select a fixed set of tests for evaluation purposes, but in both of these approaches it is unclear to what extent the resulting biased evaluation function suits its purpose.

Coevolution selects the tests used in evaluation *adaptively*; next to the population of candidate solutions, a population of tests is maintained that are used to evaluate the candidate solutions. It has been shown that such a setup can in principle yield *ideal evaluation* (De Jong & Pollack, 2004), equivalent with evaluating against all tests, using only a small set of tests. This is possible because at every point in time, only the relations between the current candidate solutions need to be evaluated. Thus, a set of tests that reveal all relations between the existing candidate solutions is sufficient.

While the theoretical possibility of ideal evaluation is interesting in itself, an important practical question is to what extent this ideal can be achieved. The DELPHI algorithm we presented in earlier work (2004) is designed to approximate ideal evaluation, and is based on the idea of searching for an ideal evaluation set as an inner loop inside the search algorithm. Since it cannot always be known whether an ideal evaluation set has already been reached, the DELPHI algorithm can only approximate ideal evaluation. Furthermore, the algorithm has a limited potential for exploration.

A second approach to achieving reliable progress in coevolution is the use of an archive that guarantees monotonic progress, i.e. progress such that a distance function exists for which the distance to the solution concept decreases with every change to the archive. The Nash Memory (Ficici, 2004; Ficici & Pollack, 2003) guarantees monotonicity for the Nash equilibrium. The IPCA algorithm (De Jong, 2004a, 2004b) guarantees monotonicity for the solution concept of the Pareto-Optimal Equivalence Set. IPCA converges to this set for any finite search space if the generator of new individuals is sufficiently explorative. Specifically, it must be capable of generating all combinations of individuals with some non-zero probability. The Pareto-optimal set is the set of solutions that provide a maximal trade-off between the different objectives, so that a solution's performance in an objective cannot be improved without reducing its performance in other objectives. Since every test is viewed as an objective in this setup, the Pareto-optimal set may be very large. An important question for the practice of coevolution therefore is whether monotonic archives for different solution concepts can be designed.

In this paper, we first define several main solution concepts that can be used in coevolution. Next, for one of these solution concepts, we design an algorithm that guarantees monotonicity, under the assumption of a finite search space. To test the degree to which this new method addresses the above limitation of IPCA, we compare the two methods in experiments. The results show that the new method offers a tradeoff between archive size and performance, and for intermediate choices of the parameter that governs this trade-off, the algorithm outperforms IPCA.

The paper is structured as follows Section 1 defines several main solution concepts for coevolution. Section 2 discusses related work. The MaxSolve algorithm is presented in Section 3. Section 4 defines the discretized COMPARE-ON-ONE problem. Results are presented in Section 5, followed by conclusions.

1. SOLUTION CONCEPTS FOR COEVOLUTION

In this section, we define several main solution concepts that can be used in coevolution.

We use the following notation. The set of all possible candidate solutions is denoted as $\mathbb{C}$, and the set of all possible tests as $\mathbb{T}$. The outcome of a test T for a candidate C may in principle come from any ordered set. Without loss of generality, it will be assumed to be a real number here. The *interaction function* used to determine this outcome is written as G (for Game): $G(C, T) \rightarrow \mathbb{R}$.

1.1 S0: Simultaneous Maximization of All Outcomes

The first solution concept requires an optimal solution C to maximize the outcome over all possible tests simultaneously, as formalized in the following requirement:

$$S0 = \{C \in \mathbb{C} | \forall C' \in \mathbb{C} : \forall T \in \mathbb{T} : G(C,T) \geq G(C',T)\}$$

This solution concept has a limited application scope, as for many problems there does not exist a single solution that simultaneously maximizes the outcome of all possible tests.

1.2 S1: Maximization of Expected Utility

The second solution concept specifies as a solution the individuals that maximize the expected score against a randomly selected opponent:

$$S1 = \{C \in \mathbb{C} | \forall C' \in \mathbb{C} : E(G(C,T)) \geq E(G(C',T))$$

where E is the expectation operator and T is randomly drawn from $\mathbb{T}$. This solution concept is appropriate for many problems, for example identifying the best chess player. It is equivalent to maximizing the sum of an individual's outcomes over all tests, or to a uniform linear weighting of the objectives. It thus implicitly assumes that all tests are of equal importance; while this may not be the case, it is a reasonable assumption in the absence of knowledge about the relative importance of the different tests.

For binary problems, this solution concept may equivalently be defined as the set of candidate solutions that solve the largest possible number of tests; the candidate solutions with the highest expected score are those that solve the largest number of tests. Thus, an alternative definition for $S1$ is:

$$\begin{aligned} S1 &= \{C \in \mathbb{C} | \forall C' \in \mathbb{C} : \\ |\{T \in \mathbb{T} | solves(C,T)\}| &\geq |\{T \in \mathbb{T} | solves(C',T)\}| \end{aligned}$$

Here, a candidate solution C is said to *solve* a test T if it has a positive outcome on the test:

$$solves(C,T) \equiv G(C,T) > 0$$

1.3 S2: Nash Equilibrium

Game theory provides the solution concept of the Nash equilibrium. A Nash equilibrium specifies a strategy for each player such that no player can profitably deviate given the strategies of the other players.

Formally, let the n classes of individuals in a problem be written as $I_1, I_2, \ldots I_n$, where for a test-based problem we could have for example $I_1 = \mathbb{C}$ and $I_2 = \mathbb{T}$. Let $I = \times_{j \in N} I_j$ where N is the set of indices: $N = \{1, 2, \ldots, n\}$. Given a set of individuals I_i, let $\Delta(I_i)$ denote the set of probability distributions over I_i, and let $\Omega = \times_{j \in N} \Delta(I_j)$. A mixed strategy profile $\alpha \in \Omega$ specifies a probability distribution for each class of individuals. The expected outcome for the i^{th} class of individuals in a mixed strategy profile is written as: $E(G_i(\alpha)) = \sum_{a \in I} \prod_{j \in N} \alpha_j(a_j) G_i(a)$, where $G_i(a)$ returns the outcome for the i^{th} individual. Then a mixed-strategy Nash-equilibrium is a mixed-strategy $\alpha*$, such that:

$$\begin{aligned} S2 = \{\alpha* \in \Omega | \forall i : \forall \alpha_i \in \Delta(I_i) : \\ E(G_i(\alpha*)) \geq E(G_i(\alpha*_1, .., \alpha*_{i-1}, \alpha_i, \alpha*_{i+1}, .., \alpha*_N)))) \end{aligned}$$

An attractive feature of the Nash equilibrium as a solution concept is that, while being general, the set of individuals it represents can be relatively small; this is a valuable property for coevolutionary search. A disadvantage is that there can be (infinitely) many Nash equilibria, part of which may be dominated; thus, finding a Nash equilibrium does not guarantee that the highest possible outcomes are achieved.

1.4 S3: Pareto-Optimal Set

Evolutionary Multi-Objective Optimization extends common evolutionary methods by facilitating the use of multiple *objectives*. This may be viewed as the use of a fitness function that is vector-valued rather than scalar. In Pareto-Coevolution, every possible test is viewed as an objective. A central concept in Evolutionary Multi-Objective Optimization is that of Pareto-dominance. Applied to the current context, a candidate solution C_1 is said to *dominate* another candidate solution C_2 if the following holds:

$$\begin{aligned} dom(C1, C2) \quad &\equiv \quad \forall T \in \mathbb{T} : G(C_1, T) \geq G(C_2, T) \\ &\wedge \quad \exists T \in \mathbb{T} : G(C_1, T) > G(C_2, T) \end{aligned}$$

The solution concept of the Pareto-Optimal Set consists of all candidate solutions that are not dominated by any other solutions:

$$S3 = \{C \in \mathbb{C} | \nexists C' \in \mathbb{C} : dom(C', C)\}$$

1.5 S4: Pareto-Optimal Equivalence Set

For each maximal combination of tests that can be solved, the Pareto-Optimal set contains all candidate solutions that solve it. Thus, the set may contain many equivalent candidates that each solve the same combination of tests. We define a variant of the Pareto-Optimal set that does not contain such duplicate candidate solutions. This set is defined by the requirement that for each maximal combination of tests that can be solved, it contains at least one candidate solution that solves it. Since multiple such sets may exist, we define $S4$ as the collection of all such sets:

$$\begin{aligned} S4 = \{S \subseteq \mathbb{C} | \forall TS \subseteq \mathbb{T} : \quad & \exists C \in \mathbb{C} : \mathit{solves}(C, TS) \\ \implies \quad & \exists C' \in S : \mathit{solves}(C', TS) \end{aligned}$$

2. RELATED WORK

A number of researchers have investigated the use of coevolution as a problem solving technique (Reynolds, 1994; Miller & Cliff, 1994; Angeline & Pollack, 1994; Schmidhuber, 1999; Pagie & Hogeweg, 1998; Paredis, 1996; Pollack & Blair, 1998; Funes, 2001; Rosin, 1997; Lubberts & Miikkulainen, 2001; Werfel, Mitchell, & Crutchfield, 2000; Moriarty & Miikkulainen, 1998; Potter & De Jong, 2000; Stanley & Miikkulainen, 2004).

While we focus on the application of coevolution to test-based problems here, coevolution can also be used to address problems where a fitness function is given; this form of coevolution is called *Cooperative* or *Compositional Coevolution* (Potter & De Jong, 2000; Watson & Pollack, 2003; Wiegand, 2003; Jansen & Wiegand, 2003)

Recent research in coevolution has benefited from an increased understanding of the adaptive search process, as expressed in the development of several recent theoretical approaches. These include the use of Evolutionary Multi-Objective Optimization concepts to describe Pareto-coevolution (Ficici & Pollack, 2000; Watson & Pollack, 2000; De Jong & Pollack, 2004), order theory (Bucci & Pollack, 2002, 2003), and game theory (Ficici, 2004; Ficici & Pollack, 2000, 2003; Wiegand, 2003; Wiegand, Liles, & De Jong, 2002).

The focus of this work is on monotonicity. In the following, we therefore discuss existing coevolutionary algorithms with a monotonicity guarantee. Rosin (1997) presents the *covering competitive algorithm*, which guarantees monotonicity for the $S0$ solution concept. Schmitt (2003) presents a convergence proof for a variant of $S0$ involving more than two types of individuals, but still requiring the existence of individuals that simultaneously maximize the outcome over all individuals of other types. Ficici has described an algorithm that guarantees monotonicity for $S2$, the Nash equilibrium. Ficici (Ficici, 2004) also discusses a different notion of monotonicity, which is applicable to open-ended spaces. Finally, the IPCA algorithm guarantees monotonicity for the solution concepts of the Pareto-Optimal Equivalence Set.

To the best of our knowledge, no monotonic method is available so far for the solution concept S1, Maximization of Expected Utility. Yet, this solution concept is of high practical relevance, as it avoids restrictive limitations (e.g. the assumption that a single solution exists that maximizes performance over all tests simultaneously), and does not have the problematic property of specifying a large number of possible solutions (for example the Pareto-front for a problem can be very large, so that the number of candidate individuals it maintains is too large for a practical search algorithm). In the following, we provide an algorithm that progresses monotonically for the solution concept S1.

```
$submit(CSnew, TSnew)\{$
  $CA := CA \cup CSnew;$
  $TA := TA \cup TSnew;$
  $\forall C \in CA$
    $no\_solved[C] = number\_solved(C, TA);$
  $\forall C_i \in CA$
    $\forall C_j \in CA, j > i$
      **if** $\forall T \in TA : G(C_i, T) = G(C_j, T)$
        $no\_solved[C_j] := 0;$
      **end**
  $sort(CA, no\_solved[]);$
  **for** $i = 1 : archive\_size$
    **if** $(no\_solved[CA[i]] > 0)$
      $select(CA[i])$
    **end**
  **end**
  $\forall T \in TA$
    **if** $\exists C \in CA : solves(C, T)$
      $select(T);$
    **end**
  $\forall T \in TA$
    $\forall T' \in TA, T' \neq T$
      **if** $\forall C \in CA : G(C, T) = G(C, T')$
        $deselect(T);$
      **end**
$\}$
```

Figure 1: Pseudocode for the MaxSolve algorithm.

3. THE MAXSOLVE ALGORITHM

We now consider how an algorithm may be devised that guarantees monotonicity for the solution concept $S1$. The algorithm is called MaxSolve, since it searches for candidate solutions that solve the maximum number of tests. Pseudocode for the algorithm is shown in Figure 1.

In order to guarantee monotonic progress towards the set of candidate solutions solving the largest number of tests, the algorithm compares candidate solutions based on how many of the tests seen by the candidate solution so far they are able to solve. By ensuring that this number can only increase, monotonicity for the solution concept of $S1$ is assured, assuming that the search space is finite.

The algorithm receives a set of new candidate solutions $CSnew$ and a set of new tests $TSnew$ that are to be considered for placement in the archive. First, it measures for each candidate how many of the tests available so far it solves. Candidates with identical outcomes for all tests are considered superfluous and assigned a zero score, so that they will not be selected.

We note that a candidate discarded in this manner could potentially solve more unseen tests than the candidates that are maintained. This does not violate the monotonicity of the archive however, since progress is measured with respect to the set of tests seen so far. If the rejected candidate solution does indeed solve more tests, this will eventually be discovered given sufficient further search, at which point it will be included.

The next step in the algorithm is to sort candidate solutions. The *sort* function sorts candidates based on the number of tests they solve. $number\_solved(C, TS)$ returns the number of tests in TS solved by a candidate solution C. Using the sorted ordering, the highest scoring individuals

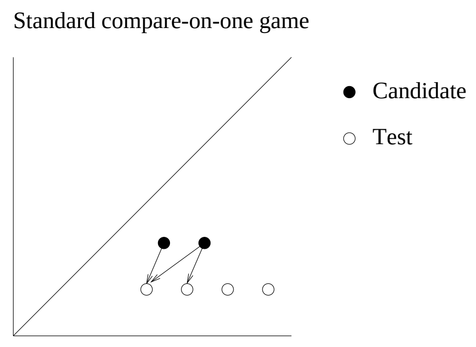

Figure 2: The standard COMPARE-ON-ONE game. Candidates are tested on the highest dimension of the test. Thus, tests below the diagonal test the horizontal dimension of candidates. Comparison of the tests solved by candidates (see arrows) provides a gradient indicating the direction of improvement.

are selected to remain in the archive, with the restriction that selected individuals must solve at least one test, and up to a maximum of n individuals, where n is a parameter specifying the maximum size of the archive.

To determine which tests to maintain, all tests solved by one or more of the selected candidates are selected. The algorithm described so far is monotonic. For efficiency, the following procedure is added: of multiple tests with identical outcome vectors only one is maintained.

The following section describes the test problem that will be used in the experiments.

4. THE DISCRETIZED COMPARE-ON-ONE PROBLEM

One of the issues that complicates the design of reliable coevolution algorithms is the problem of *over-specialization* (Watson & Pollack, 2001; De Jong & Pollack, 2004). For any problem, there can be multiple underlying factors that determine the quality of individuals, and it may well be the case that individuals only progress on one or more of these factors, but fail to do so on others.

The COMPARE-ON-ONE problem (De Jong & Pollack, 2004) is a Numbers Game problem (Watson & Pollack, 2001) that is designed to make over-specialization likely. It does so by letting tests test on a single dimension only. Both candidate solutions and tests are points in an n-dimensional space. A test tests whether a candidate is at least as high as the test in the dimension in which the test is highest:

$$m = \arg\max_i T_i$$

$$G(C,T) = \begin{cases} 1 & \text{if} \quad C_m \geq T_m \\ -1 & \text{otherwise} \end{cases}$$

where C is a candidate, T is a test, and X_i denotes the value of an individual X (C or T) in dimension i.

Due to the definition of the game, a test only evaluates a candidate on a single dimension. Unless special attention is paid to maintaining a diverse set of tests, it is unlikely that the test population will maintain tests for each dimension. This effect is enhanced by the tendency of tests to increase only in the dimension on which they test, thus moving away from the diagonal. This makes it unlikely that a lost dimension will later be regained. If a dimension is lost, individuals can only progress on some of the underlying dimensions, but will be likely to drift in others.

Individuals are supplied to the archive by a *generator*, consisting of a population of candidate solutions and a population of tests, both of size 50. The size of the new generations of candidate solutions and tests in the generator are also 50. All archives are initially empty.

To increase the difficulty of the problem, a negative *mutation bias* is used to model the property of actual problems that a variation is more likely to cause regress than progress. Thus, unless regress is avoided for all underlying objectives of the problem, the values of individuals are expected to actually *decrease* in the lost dimensions rather than just drift. The need to detect both progress and regress is further increased by applying mutation to *multiple* (two) dimensions at the same time. Specifically, mutation performs the following procedure twice: randomly select a dimension, and add a value chosen randomly uniformly from the mutation range. The mutation range is biased towards the negative side, and is [-0.15,0.1]. New individuals are generated using two-point crossover (50%) or the mutation procedure (50%).

The COMPARE-ON-ONE problem was developed to induce over-specialization. In order to determine whether overspecialization occurs, performance is measured as the *lowest* value of an individual's dimensions. Due to this, the decreasing values in a dimension will at some point result in a decreasing performance for the individual, so that over-specialization can be clearly detected. The experiments will employ the 3-dimensional version of the COMPARE-ON-ONE problem.

Since we expect exploration to be a necessary ingredient for real-world coevolutionary problems, a question is how reliability can be combined with exploration. In order to test the ability of coevolution algorithms to perform exploration, we employ a discretized version of the COMPARE-ON-ONE problem. In the discretized COMPARE-ON-ONE problem (see Figures 2, 3), the value in each dimension of an individual is rounded to the nearest multiple of a parameter δ below it. This discretization is applied to both candidates and tests before evaluating the outcome of the problem, without affecting the genotype. The ranges of the variables are unbounded, making the search space infinite; in this way an open-ended co-evolutionary process can be simulated.

$$m = \arg\max_i d(T_i) \tag{1}$$

$$G(C,T) = \begin{cases} 1 & \text{if} \quad d(C_m) \geq d(T_m) \\ -1 & \text{otherwise} \end{cases} \tag{2}$$

where $d(x) = \delta \lfloor \frac{x}{\delta} \rfloor$

As an example, using $\delta = 0.25$ as in the experiments, the individual $[0.23, 0.30, 0.47]$ would be mapped to $[0, 0.25, 0.25]$ before calculating the outcome of the standard COMPARE-ON-ONE problem. The discretization procedure greatly reduces the amount of gradient present in the problem; individuals have no means to determine whether a value of .45 is better than .25. The mutation range is set such that improvements can only be reached by making *multiple* subsequent steps in the right direction. Addressing the discretized COMPARE-ON-ONE problem therefore requires a substantial

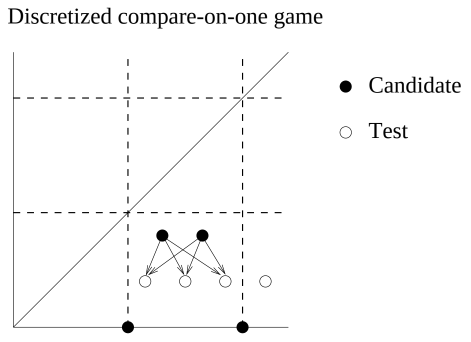

Figure 3: The discretized COMPARE-ON-ONE game. All individuals are mapped to the lower-left corner of their square in the discretization grid. Thus, the candidates solve all tests in their square. As a result, much of the gradient information is lost.

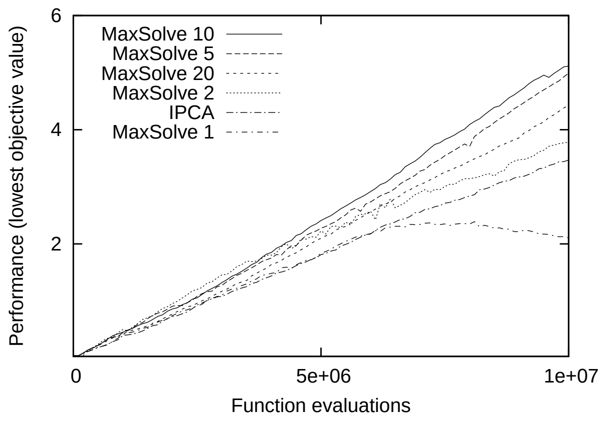

Figure 4: Experimental comparison of IPCA and Maxsolve with archive sizes 1, 2, 5, 10, and 20 on the discretized COMPARE-ON-ONE problem with mutation bias. For several parameter values, MaxSolve performs better than IPCA.

amount of random exploration in addition to the difficulties posed by the standard COMPARE-ON-ONE problem.

5. RESULTS

We now investigate the behavior of IPCA and MaxSolve on the discretized COMPARE-ON-ONE problem. Figure 4 shows the performance of the two methods measured as a function of the number of evaluations, where each evaluation computes the outcome of a test for a given candidate. All outcomes are cached, and therefore computed and measured only once. For MaxSolve, archive sizes of 1, 2, 5, 10, and 20 are used.

Performance is measured as the *lowest* dimension of an individual. This provides an objective measure of the quality of the individual. Furthermore, it has the property that the performance measure can only consistently increase if individuals are progressing in *all* underlying objectives. Thus, it is more difficult to obtain increasing performance using this measure than using e.g. the average value of the objectives.

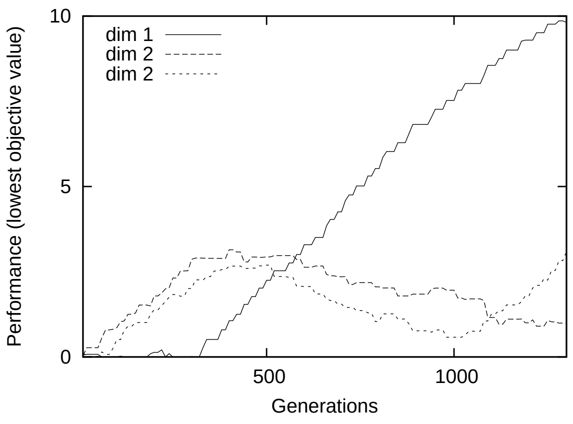

Figure 5: Performance of the best archive member in the three underlying objectives for an example run. While each transition is guaranteed to yield improvement in at least one objective, the value of the lowest objective may nonetheless decrease over time.

As the graph in Figure 4 shows, the performance of MaxSolve varies with the maximum archive size parameter. For several parameters, it performs better than IPCA. The best performance is obtained using an archive size of 10; the performance in this case is substantially higher than that of IPCA. Thus the MaxSolve algorithm is seen to provide a useful extension to the small but growing collection of reliable algorithms that are available for coevolution.

A first striking observation is that for an archive size of 1, the performance measure at some point begins to decline. This may at first sight seem at odds with the claim that the algorithm guarantees monotonicity. The reason this phenomenon can occur is that the test problem features multiple *underlying objectives* (De Jong & Pollack, 2004), as will now be discussed.

The underlying objectives for this problem correspond precisely to the dimensions of the space in which the individuals reside. Specifically, the performance of each candidate solution is determined by the three coordinates that make up its genotype. The performance measure shown in the graph reflects the *lowest* value for each individual, and shows the highest score for this value over the individuals in the archive, averaged over 30 runs.

While the individuals in the archive are selected so as to solve an incremental number of the tests that have been collected, and thus to improve in *some* underlying objective, this does not imply that the performance level obtained in each underlying objective will be maintained. This is seen when the three genotypic values of the best archive member are tracked over time; see Fig. 5. While each change in the archive improves the performance of some objective, the value of the lowest objective may decrease. As a result of this, the performance measure shown earlier can decrease, as it reflects the performance of individuals in their lowest dimension. If the problem space is finite however, the potential for improvement in each underlying objective will at some point be exhausted, after which progress in other underlying objectives is inevitable given sufficient exploration.

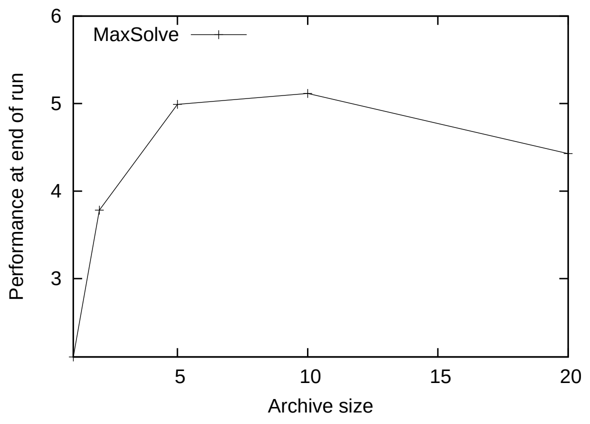

Figure 6: The trade-off between archive size and performance; the best performance is obtained for an intermediate archive size. See text.

While it is straightforward to achieve the theoretical guarantee that progress will at some point occur, it may take an impractical amount of time before such progress occurs, and accordingly the occurrence of over-specialization does pose a problem to search algorithms in practice. Therefore, it is particularly interesting to see that for larger archive sizes, the MaxSolve algorithm *is* able to avoid over-specialization. Our explanation for this observation is that given a larger number of candidates, the tests solved by the candidate archive form a more diverse set, so that individuals progressing in multiple dimensions can be successfully distinguished from those focusing on a subset of the objectives. While there is no explicit pressure to avoid over-specialization, the maintenance of a more diverse set of tests can apparently be sufficient to facilitate progress in all underlying objectives.

A further observation is that there appears to be a trade-off between the archive size and performance; the best results are obtained for an intermediate archive size. To visualize this trade-off more clearly, we have plotted the performance of MaxSolve after 10^7 function evaluations as a function of the maximum archive size. Figure 6 shows the results, and clearly shows the trade-off.

6. CONCLUSIONS

We have investigated the solution concept of maximizing the expected outcome against a random opponent. This solution concept may equivalently be seen as maximizing the number of defeated opponents or solved tests.

An algorithm that guarantees monotonicity for this solution concept has been presented, named MaxSolve. MaxSolve has been compared to IPCA, which guarantees monotonicity for the solution concept of the Pareto-Optimal Equivalence set. In experiments with a test problem likely to induce over-specialization, MaxSolve was seen to outperform IPCA for intermediate archive sizes. MaxSolve furthermore avoids the limitation of IPCA that the solution concept may involve a large number of individuals, since the algorithm maximizes the number of tests solved and avoids the maintenance of equivalent individuals. Regarding the choice of the archive size, a trade-off was observed, and intermediate values were seen to yield the best results.

Acknowledgments

The author gratefully acknowledges the anonymous reviewers for their useful comments and suggestions.

References

Angeline, P. J., & Pollack, J. B. (1994). Coevolving high-level representations. In Langton, C. G. (Ed.), *Artificial Life III*, Vol. XVII of *SFI Studies in the Sciences of Complexity*, pp. 55–71, Redwood City, CA. Addison-Wesley.

Axelrod, R. (1987). The evolution of strategies in the iterated prisoner's dilemma. In Davis, L. (Ed.), *Genetic Algorithms and Simulated Annealing*, Research Notes in Artificial Intelligence, pp. 32–41, London. Pitman Publishing.

Barricelli, N. A. (1962). Numerical testing of evolution theories. Part I: Theoretical introduction and basic tests. *Acta Biotheoretica*, *16*(1–2), 69–98.

Barricelli, N. A. (1963). Numerical testing of evolution theories. Part II: Preliminary tests of performance, symbiogenesis and terrestrial life. *Acta Biotheoretica*, *16*(3–4), 99–126.

Bucci, A., & Pollack, J. B. (2002). Order-theoretic analysis of coevolution problems: Coevolutionary statics. In *Proceedings of the GECCO-02 Workshop on Coevolution: Understanding Coevolution*.

Bucci, A., & Pollack, J. B. (2003). A mathematical framework for the study of coevolution. In *Foundations of Genetic Algorithms (FOGA-2002)*, San Francisco, CA. Morgan Kaufmann.

De Jong, E. D. (2004a). Guaranteeing progress in pareto-coevolution. In *Proceedings of the Annual Machine Learning Conference of Belgium and The Netherlands, BeNeLearn-04*, pp. 22–29.

De Jong, E. D. (2004b). The Incremental Pareto-Coevolution Archive. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-04*, pp. 525–536.

De Jong, E. D., & Pollack, J. B. (2004). Ideal evaluation from coevolution. *Evolutionary Computation*, *12*(2), 159–192.

Ficici, S. G. (2004). *Solution Concepts in Coevolutionary Algorithms.* Ph.D. thesis, Brandeis University.

Ficici, S. G., & Pollack, J. B. (2000). A game-theoretic approach to the simple coevolutionary algorithm. In Schoenauer et al., M. (Ed.), *Parallel Problem Solving from Nature, PPSN-VI*, Vol. 1917 of *LNCS*, Berlin. Springer.

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

Funes, P. (2001). *Evolution of Complexity in Real-World Domains.* Ph.D. thesis, Brandeis University, Waltham, MA.

Hillis, D. W. (1990). Co-evolving parasites improve simulated evolution in an optimization procedure. *Physica D*, *42*, 228–234.

Jansen, T., & Wiegand, R. P. (2003). Exploring the explorative advantage of the cooperative coevolutionary (1+1) EA. In Cantú-Paz, E., Foster, J. A., Deb, K., Davis, D., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R., Kendall, G., Wilson, S., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K., Jonoska, N., & Miller, J. (Eds.), *Genetic and Evolutionary Computation – GECCO-2003*, Vol. 2723 of *LNCS*, pp. 310–321, Chicago. Springer-Verlag.

Kauffman, S. A., & Johnsen, S. (1992). Co-evolution to the edge of chaos: Coupled fitness landscapes, poised states, and co-evolutionary avalanches. In Langton, C., Taylor, C., Farmer, J., & Rasmussen, S. (Eds.), *Artificial Life II*, Vol. X of *SFI Studies in the Sciences of Complexity*, pp. 325–369. Addison-Wesley, Redwood City, CA.

Koza, J. R. (1992). Genetic evolution and co-evolution of computer programs. In Langton, C., Taylor, C., Farmer, J., & Rasmussen, S. (Eds.), *Artificial Life II*, Vol. X of *SFI Studies in the Sciences of Complexity*, pp. 603–629. Addison-Wesley, Redwood City, CA.

Lindgren, K. (1992). Evolutionary phenomena in simple dynamics. In Langton, C., Taylor, C., Farmer, J., & Rasmussen, S. (Eds.), *Artificial Life II*, Vol. X of *SFI Studies in the Sciences of Complexity*, pp. 295–312. Addison-Wesley, Redwood City, CA.

Lubberts, A., & Miikkulainen, R. (2001). Co-evolving a go-playing neural network. In Belew, R. K., & Juillé, H. (Eds.), *Proceedings of the GECCO-01 Workshop on Coevolution: Turning Adaptive Algorithms upon Themselves*, pp. 14–19.

Miller, G., & Cliff, D. (1994). Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics. In Cliff, D., Husbands, P., Meyer, J.-A., & Wilson, S. W. (Eds.), *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior, SAB-94*, pp. 411–420, Cambridge, MA. The MIT Press.

Miller, J. (1989). The coevolution of automata in the repeated prisoner's dilemma. Santa Fe Institute working paper 89-003.

Miller, J. (1996). The coevolution of automata in the repeated prisoner's dilemma. *Journal of Economic Behavior and Organization*, *29*(1), 87–112.

Moriarty, D. E., & Miikkulainen, R. (1998). Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, *5*(4), 373–399.

Pagie, L., & Hogeweg, P. (1998). Evolutionary consequences of coevolving targets. *Evolutionary Computation*, *5*(4), 401–418.

Paredis, J. (1996). Coevolutionary computation. *Artificial Life*, *2*(4).

Pollack, J. B., & Blair, A. D. (1998). Co-evolution in the successful learning of backgammon strategy. *Machine Learning*, *32*(1), 225–240.

Potter, M. A., & De Jong, K. A. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, *8*(1), 1–29.

Reynolds, C. W. (1994). Competition, coevolution and the game of tag. In Brooks, R. A., & Maes, P. (Eds.), *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pp. 59–69, Cambridge, MA. The MIT Press.

Rosin, C. D. (1997). *Coevolutionary Search among Adversaries*. Ph.D. thesis, University of California, San Diego, CA.

Schmidhuber, J. (1999). Artificial curiosity based on discovering novel algorithmic predictability through coevolution. In Angeline, P., Michalewicz, Z., Schoenauer, M., Yao, X., & Zalzala, A. (Eds.), *Proceedings of the Congress on Evolutionary Computation, CEC-99*, Vol. 3, pp. 1612–1618, Piscataway, NJ. IEEE Press.

Schmitt, L. M. (2003). Theory of coevolutionary genetic algorithms. In Guo, M., & Yang, L. T. (Eds.), *Parallel and Distributed Processing and Applications, International Symposium, ISPA 2003*, pp. 285–293, Berlin. Springer.

Stanley, K. O., & Miikkulainen, R. (2004). Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, *21*, 63–100.

Watson, R. A., & Pollack, J. B. (2000). Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In Schoenauer et al., M. (Ed.), *Parallel Problem Solving from Nature, PPSN-VI*, Vol. 1917 of *LNCS*, Berlin. Springer.

Watson, R. A., & Pollack, J. B. (2001). Coevolutionary dynamics in a minimal substrate. In Spector, L., Goodman, E., Wu, A., Langdon, W., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M., & Burke, E. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-01*, pp. 702–709, San Francisco, CA. Morgan Kaufmann.

Watson, R. A., & Pollack, J. B. (2003). A computational model of symbiotic composition in evolutionary transitions. *Biosystems*, *69*(2-3), 187–209. Special Issue on Evolvability, ed. Nehaniv.

Werfel, J., Mitchell, M., & Crutchfield, J. P. (2000). Resource sharing and coevolution in evolving cellular automata. *IEEE Transactions on Evolutionary Computation*, *4*(4), 388.

Wiegand, R. P. (2003). *An Analysis of Cooperative Coevolutionary Algorithms*. Ph.D. thesis, George Mason University, Fairfax, Virginia.

Wiegand, R. P., Liles, W., & De Jong, K. (2002). Analyzing cooperative coevolution with evolutionary game theory. In Fogel, D. B., El-Sharkawi, M. A., Yao, X., Greenwood, G., Iba, H., Marrow, P., & Shackleton, M. (Eds.), *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pp. 1600–1605. IEEE Press.