DECA: Dimension Extracting Coevolutionary Algorithm

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

Anthony Bucci
DEMO Lab
Brandeis University
415 South Street
Waltham MA 02454, USA
abucci@cs.brandeis.edu

ABSTRACT

Coevolution has often been based on averaged outcomes, resulting in unstable evaluation. Several theoretical approaches have used archives to provide stable evaluation. However, the number of tests required by some of these approaches can be prohibitive of practical applications. Recent work has shown the existence of a set of underlying objectives which compress evaluation information into a potentially small set of dimensions. We consider whether these underlying objectives can be approximated online, and used for evaluation in a coevolution algorithm. The Dimension Extracting Coevolutionary Algorithm (DECA) is compared to several recent reliable coevolution algorithms on a Numbers game problem, and found to perform efficiently. Application to the more realistic Tartarus problem is shown to be feasible. Implications for current coevolution research are discussed.

Categories and Subject Descriptors

F.0 [**General**]

General Terms

Algorithms, Experimentation, Performance

Keywords

Coevolution, dimension extraction, underlying objectives, DECA

1. INTRODUCTION

Test-based coevolution algorithms (Barricelli, 1962; Axelrod, 1987; Hillis, 1990) use the outcomes of interactions between individuals to perform evaluation and selection. Historically, coevolutionary algorithms have used an aggregate measure of these outcomes during selection. For example, the score against individuals in a second population may be averaged, either directly or weighted by the number of other individuals that receive the same score as in competitive fitness sharing (Rosin, 1997). The aggregate fitness measure is then used to select individuals.

A problem with averaging over a changing set of coevolving individuals, is that this form of evaluation is unstable; features that are beneficial in the context of one set of opponents may later be unfavorable, and thus there is no guarantee that the resulting search process will improve the overall performance of individuals in the long run.

Several recent approaches to coevolution have used archives to develop a stable basis for evaluation and subsequent selection. The criterion used to decide which individuals should reside in the archive depends on the *solution concept* (Ficici, 2004) the experimenter wishes to approximate. Examples of solution concepts and corresponding algorithms include:

- Simultaneous maximization of the outcomes against all opponents, approximated by the Covering Competitive Algorithm(Rosin, 1997);
- The Nash Equilibrium, approximated by the Nash Memory (Ficici & Pollack, 2003);
- The Pareto-optimal equivalence set, approximated by the Incremental Pareto-Coevolution Archive, IPCA (De Jong, 2004a) and the LAyered Pareto-Coevolution Archive, LAPCA (De Jong, 2004b);
- Maximization of expected utility, approximated by the MaxSolve algorithm (De Jong, 2005).

The solution concept adopted in this article is the Pareto-optimal equivalence set. This concept specifies a set of candidate solutions that are *non-dominated* in the sense of evolutionary multi-objective optimization. The objectives used in determining this set are given by the set of all possible *tests*, where tests are the co-evolving individuals with which candidate solutions interact. An issue with using all tests as objectives, is that the number of possible tests in a problem is typically very large. The approach taken in this work may provide a way to reduce the set of all possible tests to a smaller set of objectives that provide equivalent evaluation.

Recent work on test-based problems has shown the existence of *underlying objectives* which compress evaluation information into a possibly small set of dimensions (De Jong & Pollack, 2004; Bucci, Pollack, & De Jong, 2004). Rather than representing the raw outcome against a single opponent, dimensions provide structure over the outcomes against many opponents. Evaluation information can then be derived from this structure. Theoretically, it has been shown

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.

that all test-based problems can be structured in this way (Bucci et al., 2004). The question then arises whether this theoretical structure can be approximated by the search algorithm, and used simultaneously to evaluate individuals. Here we begin an inquiry into that question by developing a Dimension Extracting Coevolutionary Algorithm, DECA, which extracts dimensions and utilizes this information for evaluation and selection.

DECA constitutes a novel approach to evaluation in coevolution. The algorithm constructs and updates a multi-dimensional evaluation function that aims to approximate the true underlying objectives of the problem. If this underlying multi-objective evaluation function can be approximated with sufficient accuracy, then DECA can in principle provide stable evaluation using only a possibly small subset of all tests.

To test the viability of DECA, comparison experiments with several recent reliable coevolutionary methods are performed on COMPARE-ON-ONE, a numbers game problem that is likely to induce over-specialization, and on a game called Tartarus (Teller, 1994; Ashlock, Willson, & Leahy, 2004).

The remainder of this paper is structured as follows. In Section 2, we discuss coevolutionary evaluation, Pareto Coevolution, and the notion of underlying objectives. Section 3 discusses the extraction of underlying objectives, and presents the DECA algorithm. Section 4 describes the experimental setup, and Section 5 presents results. Finally, Section 6 concludes.

2. COEVOLUTIONARY EVALUATION AND UNDERLYING OBJECTIVES

2.1 Pareto-Coevolution

A defining feature of coevolution is that the evaluation of individuals is influenced by other (co-)evolving individuals such that the *ranking* of the individuals can be affected; thus, scaling effects due order-preserving transformations are excluded.

As an example, we may imagine a population of chess players where each individual plays against some sample of the population. All individuals are evaluated based on their score against this sample, and their aim is to optimize this evaluation. This represents a first role that must be performed by a coevolutionary algorithm: individuals must perform optimization according to whatever evaluation is provided. While in regular evolution the goals of the experiments are translated directly into a fitness function, in coevolution the evaluation is provided by other, (co-)evolving individuals.

If coevolutionary optimization is to be effective, the evaluation provided by co-evolving individuals must be adequate. This implies that there is a second role for individuals in coevolution: namely, they must provide *accurate evaluation*, so that when this evaluation is optimized, the process behaves according to the goals of the experimenter. The realization that a coevolutionary algorithm must successfully address both of these two distinct roles has been an important development in coevolution research.

The value of adequate evaluation has been discussed early on (Epstein, 1994; Juillé & Pollack, 1998). An important milestone has been the suggestion that the individuals encountered in interactions, such as the opponents in two-player games, may be viewed as *objectives* in the sense of Evolutionary Multi-Objective Evaluation (EMOO). This idea is known as Pareto-Coevolution (Ficici & Pollack, 2000; Watson & Pollack, 2000), and has influenced a substantial amount of subsequent research into coevolution (Ficici & Pollack, 2001; Bucci & Pollack, 2005; Watson & Pollack, 2003; De Jong & Pollack, 2004; De Jong, 2004a, 2004b).

In correspondence with the two roles, we distinguish between two types of individuals. A *candidate solution* is an individual whose performance we wish to optimize. A *test* is an individual used to evaluate candidate solutions. In most test-based problems, there is a clear distinction between candidate solutions and tests. In Hillis's work on coevolving sorting networks for example (Hillis, 1990), the sorting networks are candidate solutions, while the sequences used to evaluated the networks (called parasites) are tests. However, individuals may also have both roles. This occurs for example in Cooperative Coevolutionary Algorithms (CCEA's); see (Bucci & Pollack, 2005).

2.2 Underlying Objectives

The notion of *underlying objectives* was first described in (De Jong & Pollack, 2004). The notion is based on the concept of objectives employed in Evolutionary Multi-Objective Evaluation. There, an objective is a function that measures some aspect of the quality of an individual, and expresses this as an element from an ordered set of scalar values. An objective can be viewed as a partial evaluation function, as it reveals some aspect of the quality of an individual; complete information about the quality of the individual is obtained by combining the information provided by all of the objectives.

In a multi-objective problem, the objectives are given as part of the problem definition. In a coevolution problem, the objectives of the problem are in general unknown. As Pareto-coevolution makes clear however, a meaningful set of objectives can be obtained by considering every possible test to be an objective. While the number of possible tests in a problem may be very large, Pareto-Coevolution has been important in providing clear candidate criteria for what a coevolutionary algorithm should be optimizing.

In the following, we describe Pareto-Coevolution and the notion of underlying objectives formally. Given a problem P, let $\mathbb{C}$ be the set of all candidate solutions, let $\mathbb{T}$ be the set of all tests, and let $G: C \times T \rightarrow \{0, 1\}$ be an interaction function that determines the outcome for candidate C on test T, which for simplicity we assume to be binary here. A test is said to *pass* a candidate if the candidate receives a positive outcome (1), and to *fail* it otherwise.

The set of objectives defined by Pareto-Coevolution will be written as O_P. Thus, for each test $T \in \mathbb{T}$, O_P contains an objective $o_T \in O_P$ that represents T. The value of an objective o_T for a given candidate solution C can simply be defined as the outcome of C on T: $o_T(C) = G(C, T)$.

A set of underlying objectives U_P can now be defined as any set of objectives that induces the same order on the candidates as O_P. Specifically, if an objective $o \in O_P$ exists for which candidate $C1$'s value is greater than candidate $C2$'s value, then U_P must contain an objective for which the same holds, and vice versa.

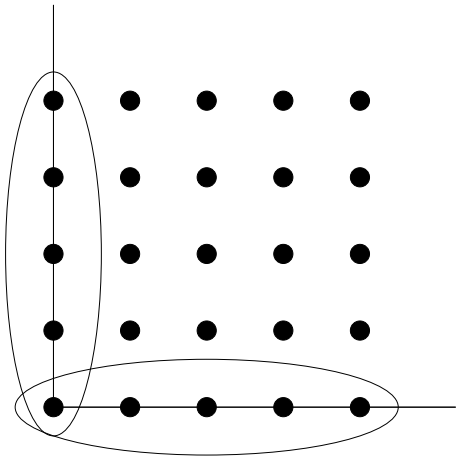

Figure 1: Example Numbers game problem: tests are points in a 2-d space. In Pareto-coevolution, all n^2 tests would be separate objectives. The two *underlying objectives* (marked by ellipses) are more compact, yet provide equivalent information.

DEFINITION 1 (UNDERLYING OBJECTIVE SET). *A set of objectives U_P is an* underlying objective set *for a problem P if and only if the following holds for all $C1, C2 \in \mathbb{C}$:*

$$\exists o \in O_P : o(C1) > o(C2) \iff \exists o' \in U_P : o'(C1) > o'(C2)$$

A set of objectives that qualifies as an *underlying objective set* provides evaluation that is precisely equivalent to using all possible tests as objectives, assuming evaluation is based on Pareto-dominance. This can be seen as follows: an individual $C1$ dominates an individual $C2$ if no objective o exists for which $o(C2) > o(C1)$, and if in addition an objective does exist for which $o(C1) > o(C2)$. Thus, the only required information to determine dominance between two individuals is whether an objective exists for which $o(C1) > o(C2)$. Since the above definition ensures this information is preserved, an underlying objective set is guaranteed to induce the same dominance relation for a set of candidates as the original Pareto-coevolution objectives of using all tests as objectives.

It can be productive to view a set of objectives as a space, which we will call the *evaluation space* of the objectives here; the evaluation of candidates then corresponds to determining its position in the evaluation space. Underlying objectives will equivalently be called *underlying dimensions*, to emphasize the geometrical aspect involved in constructing an evaluation space. To clarify the idea of underlying objectives, we will now discuss an example.

2.3 Example: A 2-dimensional numbers game

As an example, we consider a two-dimensional Numbers game (Watson & Pollack, 2001) called COMPARE-ON-ONE, which was introduced in (De Jong & Pollack, 2004). Both candidates and tests are points in two-dimensional space, see Fig. 1. In COMPARE-ON-ONE, the tests test on the dimension of their highest coordinate; tests below the diagonal base their outcome on the horizontal axis, while tests above the axis use the vertical axis to determine outcomes. The outcome of a test for a given candidate solution is obtained by determining whether the candidate solution's value in the test's selected dimension is above or below the coordinate of the test in this dimension. Thus, the game can be defined as follows:

$$\text{compare}-\text{on}-\text{one}: \quad G(C,T) = \begin{cases} 1 & \text{if} \quad C_m \geq T_m \\ 0 & \text{otherwise} \end{cases}$$

where $m = \arg\max_i T_i$, C is a candidate, T is a test, and x_i denotes the value of individual x (either candidate or test) in dimension i.

In the diagram (Fig. 1), each candidate solution is represented by a black dot. The set of tests equals the set of candidate solutions.

By adopting Pareto-Coevolution, the COMPARE-ON-ONE problem is transformed into a multi-objective problem for which each test is an objective. Since there are 5x5=25 distinct tests in this example problem, this leads to a problem with 25 objectives. However, it is clear from the description of this example problem that there are only two qualities that matter for a candidate: its horizontal coordinate and its vertical coordinate. Intuitively therefore, it would seem that the example problem should be a 2-objective problem. Consideration of the underlying objectives of the problem shows that this is indeed the case.

The following set of two underlying objectives may be defined: a horizontal objective, consisting of the tests on the horizontal axis, and a vertical objective, consisting of the tests on the vertical axis. It is straightforward to show that this set of underlying objectives satisfies the earlier definition; whenever a pair of candidates $C1$, $C2$ have different outcomes for one of the 25 tests, their coordinates on one of the axes must differ, and hence their outcome for one of the tests on the axis differs. Vice versa, if the outcomes of $C1$ and $C2$ differ for a test on one of the axes, then clearly their outcome for one the 25 tests differs, as the tests on the axes are a subset of the set of all tests.

2.4 Representation and Interpretation of Objectives

In the above example, objectives are represented by sequences of tests. For each test, we can determine the set of candidates failed by the test, called the test's Candidate Failure set, or CF-set for short, which is defined thus:

DEFINITION 2. $\text{CF}-\text{set}(T) = \{C \in \mathbb{C} \mid G(C,T) \leq 0\}$

We have employed the same notion in previous work (Bucci et al., 2004), where it was written V_t. Objectives can equivalently be represented by sequences of CF-sets. The latter provides a more powerful representation for objectives, as the tests of a problem often only represent a fraction of the set of all possible CF-sets; the latter is given by the power-set of the set of all candidates.

The objectives span an *evaluation space*, and can therefore be viewed as axes that make up a coordinate system. Each candidate and test has a *position* in the evaluation space. The position of a candidate for a given objective, which may be viewed as a coordinate on an axis, is given by the index of the highest CF-set on the axis that does not contain the candidate. This index represents the value of the objective

for the candidate. The position of a test for a given objective is given by the index of the highest CF-set that is still a subset of the test's CF-set.

2.5 Extracting Underlying Objectives

In the previous section it was shown that for an example problem, a compact set of underlying objectives existed that captures the relevant dimensions of performance in the problem. However, the choice of the underlying objectives was provided as part of the example. A more interesting question therefore is whether the underlying objectives of a problem can be extracted *automatically*. Other research demonstrates that this is possible. In earlier work (Bucci et al., 2004), we described a heuristic algorithm that accepts a matrix of outcomes, and returns a set of underlying objectives. In the algorithm, objectives are represented by sequences of tests, as in the above example.

The minimum number of objectives that form a correct set of underlying objectives for a problem is an intrinsic property of a problem, called the *evaluation dimension* of the problem. The evaluation dimension is a measure of the complexity of the evaluation function implicitly defined by a problem (Bucci et al., 2004). Using an example, it can be shown that by limiting the definition of dimensions to sequences of actual tests, the evaluation dimension of a problem may not be obtained. This is because certain combinations of outcomes can be represented using a given number of objectives only if hypothetical tests that assign a particular combination of outcomes to candidates are available; thus, limiting the definition of the objectives to combinations of outcomes actually achieved by tests may increase the required number of dimensions.

Extracting a minimal-dimensional evaluation space is possible; by considering all possible configurations of combining CF-sets onto axes for an increasing number of axes until a complete set of underlying objectives is obtained, minimal dimensionality can be achieved. However, due to its computational complexity, this approach is of theoretical interest mainly.

Here, we employ a variant of our earlier dimension extraction algorithm (Bucci et al., 2004) that, unlike that earlier algorithm, represents objectives as vectors of CF-sets rather than tests. This combines the merits of the two algorithms mentioned above; by using sequences of CF-sets as a representation for the objectives, minimality can in principle be attained. Yet, the algorithm is also computationally tractable, as it employs a heuristic to search the space of objective sets. We briefly describe this algorithm below.

3. DECA: DIMENSION EXTRACTING CO-EVOLUTIONARY ALGORITHM

3.1 Dimension Extraction Algorithm

The input to the algorithm consists of a set of n_c candidates and n_t tests, together with the full $n_c \times n_t$ matrix with outcomes of interactions between all combinations of candidates and tests. The output is a set of underlying objectives or dimensions, each of which is represented by an ordered sequence of CF-sets.

The DE algorithm operates as follows. First, the set of candidates and tests is filtered such that of any candidates or tests with identical outcome vectors, only a single one remains. The initial set of objectives is empty. The algorithm then considers whether any tests exist that fail only a single candidate. For each such candidate, an objective must exist containing a CF-set containing this candidate only; otherwise the set of objectives cannot form a complete set of underlying objectives. Since the CF-sets on an objective are supersets of the preceding CF-sets on the objective, such a CF-set can only be placed as the first element of an objective. Therefore, a separate objective is constructed for each of the above candidates.

Next, the following cycle is repeated. For each test, the current value for each objective is determined as the highest CF-set on the objective for which the test still fails all candidates. Starting from the set of candidates failed by each test, the candidates occurring in the highest CF-set of the test are removed. The remaining candidates failed by the test (remaining Candidate Failures, or CF's) are yet to be accounted for by the coordinate system.

The collection of all remaining CF's is sorted based on the number of tests that require the CF's. The algorithm first attempts to place one of these remaining CF's, visited in the sorted order, at the end of an objective for which it holds that all test that have the CF as a remaining CF have all CF's specified by the highest CF-set on the objective. This guarantees that by appending the CF to the objective, all tests that still needed the CF can now remove it from their list of remaining CF's. If this fails for all CF, the algorithm attempts to add a CF to an objective such that not all, but the highest possible number of tests, but at least one test, benefit from adding the CF. If this fails for all remaining CF's as well, a new objective is created, on which the most frequently required CF is placed.

The above cycle is repeated until all CF's made by tests are accounted for by the current set of objectives, i.e. no test has any remaining CF's. The DE algorithm returns the set of objectives ($objs$), and a corresponding set of test objectives ($testobjs$). For every CF-set on an objective $obj \in objs$, every test whose position on obj equals the CF-set is placed on the corresponding $testobj$.

So far, we have discussed the notion of underlying objectives, and briefly described a Dimension Extraction (DE) procedure for extracting a set of underlying objectives for a given set of candidates and tests. The significance of this procedure is that the resulting objectives define an evaluation space that can be used to evaluate candidate solutions. Given the complete set of candidates and tests for a problem, the DE procedure constructs a set of objectives whose values together specify all relevant evaluation information for a candidate solution. Thus, evaluating the candidate solution based on the extracted objectives is equivalent to evaluating the candidate on all tests, and thereby provides maximally informative evaluation.

Rather than applying the DE algorithm to the complete sets of candidates and tests, the idea behind DECA is to build the evaluation space gradually. Starting from an empty evaluation space, the algorithm receives sets of candidates and tests, and calls the DE procedure to construct an evaluation space that provides complete evaluation information for the current set of candidates and tests. Next, for all axes on the objectives returned by the DE procedure, DECA retains the candidates represented by the CF-sets on the objectives. In addition, all tests placed on the test objectives by the DE procedure are kept. By keeping these sets of candidates

and tests, the next approximation of the evaluation space is guaranteed to represent at least the same evaluation information.

It can be shown that the above description of DECA is guaranteed to converge to a complete underlying objective set, as will now be discussed. To show this, we consider the set of *distinctions* (Ficici & Pollack, 2001) made by subsequent versions of the evaluation space. An objective o makes a distinction between candidates $C1$ and $C2$ if it assigns a higher value to the former than to the latter: $o(C1) > o(C2)$. It will be shown that the set of distinctions made for the set of candidates retained by DECA grows over time.

Assume a test T and a pair of candidates $C1, C2$ exist such that $G(C1, T) > G(C2, T)$. Since G is assumed to be binary, this implies T fails $C2$ but not $C1$. In the evaluation space E returned by the DE procedure, T's position is represented by a vector of CF-sets that represent coordinates on the dimensions of E. The union of these CF-sets equals the set of all current candidates failed by T. Since T fails $C2$, at least one of these CF-sets must contain $C2$, and since T does not fail $C1$, this CF-set cannot contain $C1$. Given that for each position on a test objective, all tests that have the position are retained, we know that T will be retained. This guarantees that in the next approximation of the evaluation space, and by induction in all subsequent versions of the evaluation space, a test T making the distinction between $C1$ and $C2$ will again be present. Therefore, the set of distinctions made by the evaluation space can only grow.

The above shows that the set of distinctions made can only grow. Furthermore, any distinction that can be made between two candidates can be incorporated into DECA's evaluation space, as long as it is presented to the algorithm. Therefore, to guarantee that the algorithm will converge to an evaluation space representing all possible distinctions, it is sufficient to ensure that all combinations of two candidates and a test will be presented to the algorithm with a non-zero probability. This can be ensured theoretically by generating individuals uniformly random (though in practice this method of finding missing distinctions may not be the most efficient choice). If it can further be assumed that the spaces of candidates and tests are finite, then the algorithm must converge to an evaluation space that represents all possible distinctions. Definition 1 shows that such an evaluation space is an underlying objective set.

The above version of DECA is described to motivate the theoretical idea behind the algorithm. However, the sets of candidates and tests maintained by the above procedure can still be large. Therefore, two changes are made to define the implementation of the algorithm. First, rather than maintaining *all* tests that have a given coordinate on a test objective, the algorithm keeps only one, namely the test that fails fewest other candidate solutions. Furthermore, if candidates are progressing on all objectives, it is sufficient to maintain only the high end of each objective, representing the coordinates of the best current candidates. Therefore, only the highest `max-elements` elements of each axis are maintained. The resulting version of DECA is described in pseudo-code in Figure 2.[1]

An important practical question is whether the evaluation spaces of practical problems can be compactly described. The example in the previous section showed that this is at least possible. Stronger evidence to suggest this was found in a preliminary investigation of the game of Nim; we found that a complete space of 36 distinct tests could be reduced to only four underlying objectives, representing a significant degree of compression.

[1]The notation $f(\&x)$ denotes call-by-reference, i.e. the function may change the argument x.

```
deca(Cnew, Tnew){
  Cset = ∅
  Tset = ∅
  initialize(Cpop)
  initialize(Tpop)
  while(!done)
    evolve(Cset, Tset, &Cpop, &Tpop, &Cnew, &Tnew)
    extract(Cset ∪ Cnew, Tset ∪ Tnew, &objs, &testobjs)
    Cset := {}
    Tset := {}
    ∀obj ∈ objs
      n := obj.length()
      for i := max(0, n − max − elements) ... n − 1
        CF := obj[i] \ obj[i − 1]
        Cset := Cset ∪ CF
    ∀obj ∈ testobjs
      n := obj.length()
      for i := max(0, n − max − elements) ... n − 1
        Tset := Tset ∪ arg min_{T∈obj[i]} |CF − set(T)|
  end
}
```

Figure 2: Pseudo-code of the DECA algorithm. The algorithm receives sets of new candidates and tests, and applies the Dimension Extraction algorithm (DE), resulting in a set of underlying objectives. For each objective, and for up to `max-elements` of the highest CF-sets on the objective, the corresponding candidate and a test whose position on the objective corresponds to the CF-set (if possible) are maintained. In this way, increasingly high sections of the underlying objectives, and corresponding candidates, are maintained.

3.2 Evaluation in DECA

DECA extracts objectives, which represent vectors of candidates and tests. In the basic setup, the candidates and tests on the objectives are collected, and the resulting sets of candidates and tests are used to evaluate population individuals, and potentially to generate new individuals, both without using information about the positions of the individuals on the objectives.

A crucial feature of DECA however is that the extracted objectives define an evaluation space that can be used to accurately evaluate individuals. The objectives can be used in several ways to perform evaluation.

A first, straightforward idea is that since we have access to multiple objectives, any algorithm from the growing field of Evolutionary Multi-Objective Evaluation can be applied. We test this idea by sorting individuals based on the number of individuals by which they are dominated. The objectives of individuals are given by their positions on the objectives, as defined above. This algorithm is called DECA-MO.

As second idea is to aggregate the objective values of individuals. We test several possibilities. DECA-SUM sums the

values of the objectives, and uses this score for selection. DECA-MIN calculates the minimum objective value; if this value is high, the individual has a high value for all objectives. DECA-MINSUM first sorts by the minimum value of the objectives of the individual, and for equal values lexically sub-sorts based on the sum of the objective values. Finally, DECA-MINSUMAVG uses the average of the minimum and sum of the objective values as a score.

4. EXPERIMENTAL SETUP

We now investigate DECA in experiments. The various ways to use the evaluation information provided by DECA are compared, and DECA is compared to other coevolutionary algorithms.

4.1 Comparison Algorithms

DECA is compared to three coevolutionary archive methods: IPCA, LAPCA, and MaxSolve. Furthermore, to compare with standard coevolution algorithms, we compare two non-archive setups, called STANDARD and ADVANCED. DECA and the above archive methods used ADVANCED to supply new individuals; this algorithm is described below. The candidate and test populations are both of size 20 in all experiments. The outcomes of interactions between candidates and tests are cached to avoid unnecessary evaluations. For each methods, 10 runs are performed, and the curves show the average performance over these runs, unless otherwise specified.

4.1.1 Generation of New Individuals

For the non-archive methods, candidate parents are selected from the candidate population, and test parents from the test population. For DECA and the archive methods, candidate parents are randomly selected from the population, while tests are selected from the union of the test population and the tests maintained by DECA or the test archive.

For the COMPARE-ON-ONE problem, individuals are generated by mutation only. Mutation there randomly selects two dimensions, and adds a value selected randomly uniformly from [-0.06,0.04]; the negative mutation bias and the fact that two dimensions are mutated at once render the problem difficult, as detecting regress in one of the dimensions requires accurate evaluation.

For Tartarus, the genetic representation and operators of variation are as described in (Ashlock et al., 2004). Candidates are generated by mutation, and tests are generated by generating random new boards.

4.1.2 Evaluation and Selection

Candidates and test are assigned scores as follows. Each positive outcome of a candidate against a tests counts as one point, and vice versa. For DECA and the archive methods, candidates are evaluated on the tests in the population and those maintained by DECA or the archive, and vice versa for tests. If distinction objectives are used (ADVANCED), each pair of candidates between which a test makes a distinction (Ficici & Pollack, 2001) contributes a point. If fitness sharing is used (ADVANCED), each point is divided by the number of other individuals that make the same point (a positive outcome or a distinction); otherwise (STANDARD), each point contributes a value of one. The outcomes of interactions and the number of distinctions made are weighted as 3:1, and then summed, yielding an overall score by which

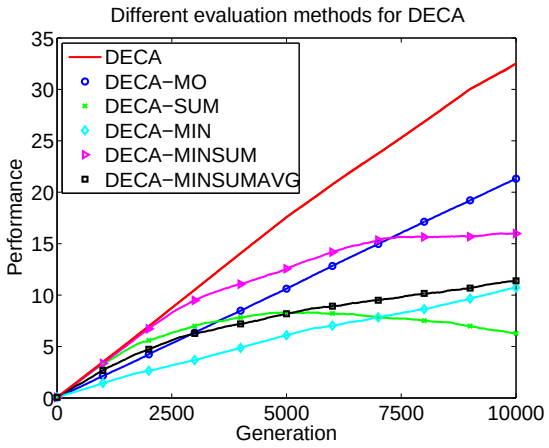

Figure 3: Comparison of DECA and variants.

the individuals are ranked. If duplicates are discounted (ADVANCED), of any individuals with identical outcome vectors, all but one are assigned zero overall scores. The highest scoring POPSIZE individuals are selected.

4.1.3 Archive Methods and DECA

Individuals are only submitted to DECA or the archive if they are not yet stored there. DECA and the archives of archive methods are updated every 10 generations. For DECA, only the highest `max-elements` = 10 elements CF-sets on each axis are maintained. The archive size for MaxSolve was set to the average archive size observed for DECA in the COMPARE-ON-ONE experiments: 25. For LAPCA, six different numbers of layers were used: 1, 2, 5, 10, 20, and 50.

4.2 The Tartarus Problem

In addition to the three-dimensional COMPARE-ON-ONE problem that was described above, a more realistic problem called *Tartarus* (Teller, 1994) is used in the experiments. Tartarus is a standard test problem which has been used in other coevolution work (Ashlock et al., 2004), and is described in detail there. Briefly, Tartarus is a board game where bulldozers move boxes around on a 2-d board. The aim is to place box sides against the walls. Candidates are GP automata that encode a strategy for the game, and tests are randomly generated boards. In the remainder of this section, we describe the algorithms that are compared.

4.3 Performance measures

The performance of individuals in COMPARE-ON-ONE can be measured objectively. The performance criterion is to maximize the *lowest* of the three values that define an individual. Thus, the performance of individuals can only increase if the values of individuals are increasing in *all* three dimensions. In Tartarus, the performance of a candidate is its score on a fixed set of 100 randomly generated boards. Experiments are run for 10,000 generations.

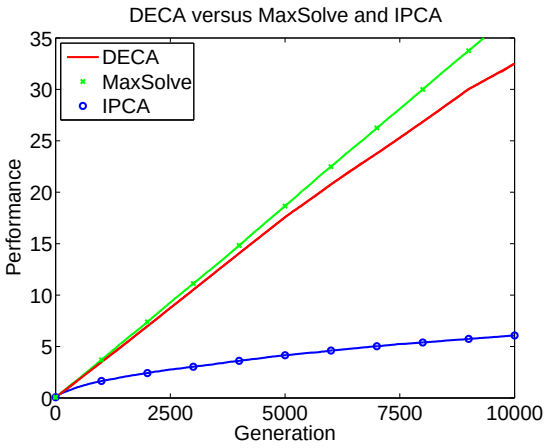

Figure 4: DECA versus MaxSolve and IPCA.

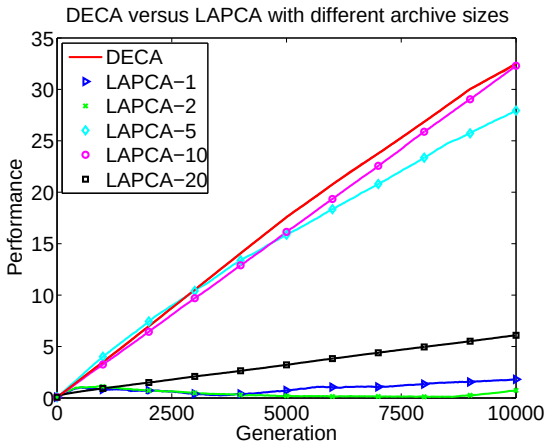

Figure 5: DECA versus LAPCA.

5. RESULTS

5.1 Results for Compare-on-one

In this section, we report experimental results with DECA and the comparison methods. First, we compare the different variants of DECA; see Figure 3. The graph shows that the standard version of DECA performs best. In this algorithm, the candidates and tests on the objectives are collected and the resulting sets are used for evaluation. It is striking that standard DECA outperforms the DECA-MO version, which uses multi-objective selection. Our explanation for this is that the evaluation method used by the population uses distinctions and fitness sharing.

The aggregate methods (DECA-MIN, DECA-SUM, DECA-MINSUM, and DECA-MINSUMAVG) all have a substantially lower performance than DECA and DECA-MO. Therefore, in further experiments, DECA is employed.

Figure 4 compares DECA to MaxSolve and IPCA. While MaxSolve performs best, DECA performs comparably, and both methods clearly outperform IPCA. The latter is not surprising; while IPCA is a reliable coevolution method, its test population grows quickly. The resulting slowdown of the algorithm is visible as a slight decline in the slope of the curve.

Figure 5 compares DECA with LAPCA. The best performance for LAPCA is obtained with 5 or 10 layers. DECA performs comparably to or better than these best instances of LAPCA.

The standard, non-archive coevolutionary methods (STANDARD and ADVANCED) were also run on the COMPARE-ON-ONE problem, but for both methods the average performance remained below one, and it is therefore not plotted; the COMPARE-ON-ONE problem can only be addressed by coevolutionary methods that perform accurate evaluation or use other techniques to ensure stable progress.

In summary, on the COMPARE-ON-ONE problem, DECA performs comparably to the best comparison methods, demonstrating the potential of the method to establish reliable evaluation using limited resources.

5.2 Results for Tartarus

Finally, we ran DECA, MaxSolve, and the non-archive methods on the slightly more realistic Tartarus problem; see Fig. 6. It is encouraging that DECA, a theoretically founded algorithm, can feasibly be run on a test problem that is closer to practice than more artificial test problems such as COMPARE-ON-ONE, and achieve reasonable performance. However, it is also striking that on this problem DECA performs no better than the much less complex non-archive coevolutionary methods STANDARD and ADVANCED.

This suggests that difficulties which reliable coevolution methods are designed to overcome play only a minor role in Tartarus. This finding leads to a new question for current coevolution research: which problem features are most influential in limiting performance on practical coevolution problems?

6. CONCLUSIONS

We have described DECA, a coevolutionary algorithm that extracts the underlying objectives of a problem and simultaneously uses these to evaluate individuals. The number of tests on the dimensions of an evaluation space can be vastly less than the total number of tests. Due to this reduction, evaluation in DECA can be efficient and yet provide accurate and reliable information.

The novel approach to evaluation that has been investigated was compared with several recent reliable coevolution methods. On the COMPARE-ON-ONE problem, the algorithm was found to perform comparably to the best comparison methods. The algorithm was also applied to the Tartarus problem. The application to this more realistic problem demonstrates that it is feasible to bring the ideas embodied by DECA into practice. However, while DECA and MaxSolve are based on theoretical principles, no substantial performance improvements were observed compared to more basic comparison methods. Thus, while theoretical research into coevolution has certainly improved insight into certain problems that can occur in coevolution problems, and has yielded solutions to several of these problems, the results that were obtained pose a new question for current coevolution research; namely, to identify features of coevolution problems of practical interest that limit the performance of

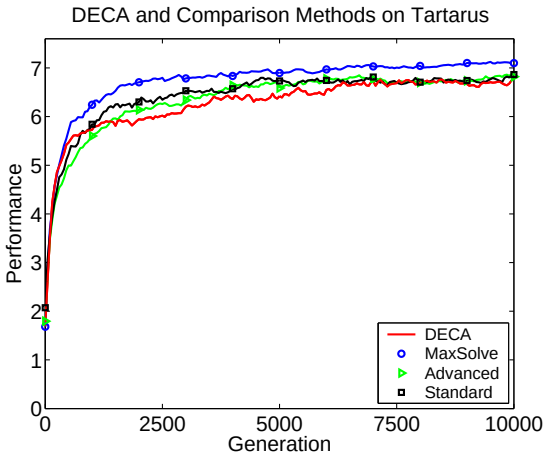

Figure 6: Performance of DECA, MaxSolve, and two standard coevolution methods on the Tartarus game.

current coevolution methods. We hope the extraction of underlying objectives that has been employed here may serve as a useful tool in analyzing the unknown complexities of these problems.

In contrast to algorithms inspired by biological principles, DECA derives from a theoretical understanding of problem structure. We take the performance of this algorithm as evidence that its theoretical grounding is both sound and useful. A goal of future work is to further this theoretical analysis to the point that we can produce algorithms which demonstrably outperform existing algorithms.

7. ACKNOWLEDGEMENTS

The authors wish to thank Dan Ashlock for kindly providing his Tartarus code.

References

Ashlock, D., Willson, S., & Leahy, N. (2004). Coevolution and tartarus. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, pp. 1618–1624, Portland, Oregon. IEEE Press.

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.

Bucci, A., & Pollack, J. B. (2005). On identifying global optima in cooperative coevolution. In Beyer, H.-G., et al. (Ed.), *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, Vol. 1, pp. 539–544, Washington DC, USA. ACM Press.

Bucci, A., Pollack, J. B., & De Jong, E. D. (2004). Automated extraction of problem structure. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-04*, pp. 501–512.

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

De Jong, E. D. (2004b). Towards a bounded Pareto-Coevolution archive. In *Proceedings of the Congress on Evolutionary Computation, CEC-04*, pp. 2341–2348.

De Jong, E. D. (2005). The MaxSolve algorithm for coevolution. In Beyer, H.-G. (Ed.), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-05*, pp. 483–489. ACM Press.

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

Epstein, S. L. (1994). Toward an ideal trainer. *Machine Learning*, *15*(3), 251–277.

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, M., et al. (Ed.), *Parallel Problem Solving from Nature, PPSN-VI*, Vol. 1917 of *LNCS*, Berlin. Springer.

Ficici, S. G., & Pollack, J. B. (2001). Pareto optimality in coevolutionary learning. In Kelemen, J. (Ed.), *Sixth European Conference on Artificial Life*, Berlin. Springer.

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

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

Juillé, H., & Pollack, J. B. (1998). Coevolving the "ideal" trainer: Application to the discovery of cellular automata rules. In *Proceedings of the Third Annual Genetic Programming Conference*.

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

Teller, A. (1994). The evolution of mental models. In Kinnear, K. E. (Ed.), *Advances in Genetic Programming*, Complex Adaptive Systems, pp. 199–220, Cambridge. MIT Press.

Watson, R. A., & Pollack, J. B. (2000). Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In Schoenauer, M., et al. (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., et al. (Ed.), *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.