Towards a Bounded Pareto-Coevolution Archive

Edwin D. de Jong
Decision Support Systems Group
Institute of Information and Computing Sciences
Utrecht University
PO Box 80.089
3508 TB Utrecht
The Netherlands
E-mail: dejong@cs.uu.nl

***Abstract*— Coevolution offers adaptive methods for the selection of tests used to evaluate individuals, but the resulting evaluation can be unstable. Recently, general archive-based coevolution methods have become available for which monotonic progress can be guaranteed. The size of these archives may grow indefinitely however, thus limiting their application potential. Here, we investigate how the size of an archive for Pareto-Coevolution may be limited while maintaining reliability. The LAyered Pareto-Coevolution Archive (LAPCA) is presented, and investigated in experiments. LAPCA features a tunable degree of reliability, and is found to provide reliable progress in a difficult test problem while maintaining approximately constant archive sizes.**

I. Introduction

For many problems, candidate solutions (*learners*) are evaluated based on their performance on *tests*; examples include concept learning, function approximation, and game playing. If the number of tests for such *test-based problems* is high, evaluation has to be based on a selection of tests, but the identification of an informative subset of the tests can be a difficult optimization problem in itself.

Coevolution offers an approach to adaptively select tests for the evaluation of learners [1], [2], [3], [4], [5], [6]. Since an adaptive test set can render evaluation unstable, an important question is how coevolution may provide stable progress. Examples of unstable evaluation include over-specialization, Red Queen dynamics, and disengagement [7], [8]. While coevolution has produced interesting results in spite of these issues [1], [9], [10], [11], [12], [13], most coevolution algorithms that have been used in practice so far are heuristic in nature and do not offer a guarantee that progress will be made over time.

Several algorithms are available that do provide a guarantee of progress. Progress will here be taken to mean that regress can be avoided; if this can be achieved, then the combination with a generator of individuals that occasionally makes improvements results in a progress guarantee. A first example is Rosin's *covering competitive algorithm* [14], which requires a new learner to defeat all tests in the archive, and vice versa. The *dominance tournament* [15] can be seen as a specific case of the covering competitive algorithm. While introduced as a method for tracking progress in coevolution, the method can also be used as a coevolutionary archive [16]. Finally, Schmitt [17] presents a stochastic model aimed at demonstrating convergence to global optima with the requirement that for at least some species, strictly dominant individuals exist that maximize performance over all possible evaluation environments.

A serious obstacle in applying the above algorithms to test-based problems in practice, is that the class of problems to which they are applicable is highly restricted. For most solution concepts of interest, progress to global optima can only be guaranteed by these methods in problems where a single learner exists that solve all solvable tests. This requirement is very strict, and for many test-based problems of practical interest it does not hold. It is a desirable goal therefore to develop archive methods that are more *general*, in the sense that they guarantee progress for a broader class of problems.

Recently, two algorithms have been introduced that are more general, and guarantee progress in coevolution without requiring the existence of individuals that defeat all previous opposition. The *Nash Memory* [16] guarantees progress for the game-theoretic solution concept of the Nash equilibrium. The *Incremental Pareto-Coevolution Archive* [18] guarantees progress for the solution concept of the Pareto-front. For both archives, the guarantee of progress requires an unbounded archive size.

The guarantee that no regress can be made is a valuable property of coevolution archives. However, if the size of an archive cannot be limited, the computational costs of evaluation may increase over time. An important question with respect to the practical application of coevolution therefore is how archive size may be limited while still providing reliability. The Nash Memory can be pruned to limit the archive size [16], and this may provide a practical method for symmetric coevolution problems using the Nash equilibrium as a solution concept. Since pruning is random however, it is unclear how reliability is affected by limiting the size of the archive.

We investigate to what extent the size of a reliable archive may be limited while maintaining reliability. We describe a layered variant of the Incremental Pareto-Coevolution Archive (IPCA) with a tunable number of layers. Increasing this parameter improves reliability at the expense of a larger archive size. This provides a tunable degree of accuracy, so that reliability may be maximized given the available compu-

tational resources. The effect of the number of layers on both performance and archive size is investigated in experiments, and limitations are discussed.

The structure of the paper is as follows. First, we briefly describe Pareto-Coevolution (Section II) and the notion of underlying objectives. The Incremental Pareto-Coevolution Archive is described in Section III. Next, in Section IV, we explore the effects of maintaining a limited number of layers. This leads to a layered variant of IPCA called the Layered Pareto-Coevolution Archive (LAPCA) described in Section V. The experimental setup is described in Section VI. Section VII reports experimental results, followed by conclusions.

II. Pareto-Coevolution

Coevolution has long been seen as the simultaneous adaptation of individuals whose fitness measures are interdependent. While this view is correct and serves to define the domain of coevolution problems, it does not make clear how coevolution could be made to provide reliable progress, or even what progress would mean, precisely because the fitness measure depends on other evolving individuals. The paradigm of Pareto-Coevolution [19], [20] offers a coherent view of coevolution by defining a precise criterion for the evaluation of learners. Specifically, Pareto-Coevolution proposes that candidate solutions (*learners*) are to be evaluated by viewing the outcomes of their interactions with other individuals called *tests* as *objectives* in the sense of Evolutionary Multi-Objective Optimization (EMOO).

As a direct consequence of defining the objectives of learners as the outcomes of the tests, the optimal solution of a Pareto-Coevolution problem is the set of all learners that are *non-dominated* with respect to these objectives. A learner A dominates another learner B if A's outcomes are all at least as high as B's, and in addition at least one outcome is higher then B's outcome for the corresponding test:

$$\begin{gathered}\forall T \in \mathcal{T}: \quad G(A,T) \geq G(B,T) \\ \wedge \\ \exists T \in \mathcal{T}: \quad G(A,T) > G(B,T)\end{gathered}$$

where $\mathcal{T}$ is the set of all possible tests and $G(L,T)$ is the outcome of learner L on test T. The set of all non-dominated individuals is called the *Pareto-front*, and forms the *solution concept* [21] of Pareto-Coevolution.

A. Underlying Objectives

The set of objectives in Pareto-Coevolution is given by the set of all possible tests. This set may be very large or infinite. For many test-based problems however, a smaller set of objectives that provides equivalent information can be defined.

The *underlying objectives* or underlying dimensions [22], [6] of a test-based problem can be defined as follows. Let a *dimension* be an ordered sequence of tests, such that any learner failing a test fails all subsequent tests. For any such sequence, knowledge of the first test T_i failed by a learner uniquely specifies the outcomes of that learner for all tests in the dimension, namely a test T_j is failed if and only if $j \geq i$. Given the notion of a dimension, we can define a coordinate system as a set of dimensions such that each test is represented in some dimension. The *underlying objectives* of a problem can now be defined as a set of dimensions of minimal cardinality that forms a coordinate system for the problem. The notion of a coordinate system, and a preliminary algorithm for extracting the underlying objectives of a test-based problem, are discussed in detail in [23].

III. The Incremental Pareto-Coevolution Archive

In this section we briefly describe the Incremental Pareto-Coevolution Archive (IPCA) introduced in [18]. IPCA consists of a learner archive and a test archive. The algorithm provides procedures to decide which newly generated learners and tests will enter the archive. The learner archive is periodically updated to maintain non-dominated learners only.

The algorithm operates as follows. A newly generated learner is *useful* with respect to a set of learners LS and a set of tests TS if it is not dominated by any learner in LS, and if there is no learner in LS which has equal outcomes for all tests in TS:

$$\begin{gathered}useful(L, LS, TS) \; = \\ \nexists L' \in LS: \; L' \overset{TS}{\succ} L \quad \wedge \\ \nexists L' \in LS: \; \forall T \in TS: G(L,T) = G(L',T)\end{gathered}$$

A related function called $useful-tests(TG, T^t, LG, L^t)$ identifies tests in a new generation of tests TG that are required in addition to T^t in order to detect that certain learners in LG are *useful* with respect to L^t. Specifically, if a learner is not useful based on T^t but is useful based on $T^t \cup T_1, T_2, \ldots T_k \in TG$, then $T_1, T_2, \ldots T_k$ contains useful tests and a subset of TG with this property will be returned by $useful-tests$. Additionally, if for any learner $L \in LG$, there is a test $T \in TG$ that defeats the learner and the learner is not defeated by any test in T^t, then L and T are marked as *useful*. Using these functions, the IPCA algorithm can be described as shown in figure 1.

A. Monotonicity and Convergence

The operation of any archive inevitably depends on the new individuals provided by the generator. The criterion required of a coevolution archive is therefore that progress can be guaranteed given the arrival of new individuals which occasionally represent progress. The latter can be guaranteed by generating every possible individual with a non-zero probability.

A proof that the IPCA algorithm guarantees monotonic progress is provided in [18]. If the number of different learners and tests is finite and all learners and tests are generated with non-zero probability, then the property of monotonic improvement implies convergence to the global optimum of the Pareto-front over all possible tests.

$L^0 := \emptyset$
$T^0 := \emptyset$
$t := 0$
while $\neg done$
  $L^t := non-dominated(L^t, T^t)$
  $L^{t+1} := L^t$
  $T^{t+1} := T^t$
  $LG := generate-learners(L^t)$
  $TG := generate-tests(T^t)$
  $TS := useful-tests(TG, T^t, LG, L^t)$
  $T^{t+1} := T^{t+1} \cup TS$
  for $i = 1 : |LG|$
    if $useful(L_i, L^{t+1}, T^{t+1})$
      $L^{t+1} := L^{t+1} \cup L_i$
  end
  if $L^{t+1} \neq L^t$
    $t := t + 1$
end

Fig. 1. The Incremental Pareto-Coevolution Archive (IPCA). Monotonic progress is guaranteed for this archive-based Pareto-Coevolution algorithm [18].

IV. Limiting the Archive Size

While IPCA maintains only a single layer of learners, the size of the test archive can grow indefinitely over time. In the following, we investigate how archive size may be limited while still providing reliability as much as possible.

In general, the aim of a Pareto-Coevolution archive is to maintain high performance learners and informative tests. However, if only high performance learners are maintained, the capacity to determine which tests are informative may be lost. This in turn affects the ability to distinguish high quality learners from others. Because of this mutual dependency, the learner and test archives not only serve to maintain valuable individuals, but must also function as scaffolds for one another. We present archive update procedures that permit limiting the size of the archives while achieving a tunable degree of reliability.

The method that will be described maintains a limited number of layers of learners, and tests that separate these layers from one another. For such a layered archive, the archive sizes are bounded in terms of the sizes of the layers of learners. While the latter may still be large for problems with continuous objective functions, it is finite for any test-based problem with a finite number of underlying objectives and distinct tests per objective. Below, we describe the criteria for acceptance of individuals into the archive in detail.

A learner archive is partitioned into layers, as done in the NSGA algorithm for EMOO [24]. First, all non-dominated learners are removed from the archive. These form the first layer. Next, from the remaining learners, we select the learners that are non-dominated once the first layer has been removed; these learners form the second layer. Continuing in this manner, a set of up to n layers can be obtained for a chosen n.

	T1	T2
L1	0	0
L2	1	0
L3		1

Fig. 2. One-layer example leading to regress. The table lists the outcomes for the learners.

The subdivision into layers provides a method to select which learners from the learner archive will be maintained. The maintenance of the learner archive requires the presence of certain tests; if important tests are missing, the qualities of certain learners may not be detected when a second iteration of the learner archive update procedure is performed, resulting in the loss of those learners.

The requirement on the test archive is therefore that it must *separate* the layers of learners from one another, so that the layering of the learning archive will be preserved in subsequent learner update steps. To this end, given two subsequent layers of learners, we define a *separation set* as follows. For any learners L_i, L_j in layers i and j where $i = j$ or $i = j - 1$, if any available test makes a *distinction* between L_i and L_j, then the separation set must contain a test that makes this distinction. A test makes a distinction [25] between learners L_i and L_j if it assigns a higher score to L_i than to L_j:

$$dist(T, L_1, L_2) \iff G(L_1, T) > G(L_2, T)$$

A. Reliability for 1, 2, and 3 layers

To study the effect of the number of layers maintained on the reliability of the archive, we consider cases where a small number of layers is kept and construct examples where the reliability of the archive cannot be provided.

1) One layer: Figure 2 shows an example where only a single layer of learners is kept. Table I specifies in detail which learners and tests are present in the archive over time. In addition, individuals which have just been generated are shown. In the first generation (G1), learners L_1 and L_2 and test T_1 are supplied to the archive.[1] Since L_2 dominates L_1 when the set of available tests consist of T_1, only L_2 is maintained in the first update step (U1) given that only a single layer of learners is maintained. Now that L_1 has been removed, T_1 no longer makes a distinction between any learners, and is thus also removed. Next, L_3 and T_2 arrive. Since L_3 dominates L_2 given only the presence of T_2, L_2 is removed. Since there is no guarantee that L_3 solves T_1, this is a potential instance of regress. We note that while it would be possible to simply keep all tests that have been solved so far, this would not lead to an archive of bounded size.

[1]While learner L_1 is not strictly necessary to demonstrate potential regress, we wish to show an example where a test (T_1) has already been found to be useful but is yet discarded.

	G_1	U_1	G_2	U_2
L_1	+			
L_2	+	+	+	
L_3			+	+
T_1	+			
T_2			+	

TABLE I

PRESENCE OF LEARNERS AND TESTS OVER TIME WHEN ONLY THE FIRST (NON-DOMINATED) LAYER OF LEARNERS IS KEPT. G_i REPRESENTS GENERATION i, U_i REPRESENTS THE SITUATION AFTER UPDATE STEP i. A '+' INDICATES THE PRESENCE OF A LEARNER OR TEST AT A GIVEN POINT IN TIME.

	T1	T2	T3	T4
L1	0	0	0	
L2	1	0	0	0
L3	1	1	0	0
L4		1	1	0
L5			1	1

Fig. 3. Two-layer example leading to regress.

2) Two layers: The previous example showed that the maintenance of a single layer of non-dominated learners can be insufficient. If only a single layer of learners is maintained, there is no way to determine which tests separate these non-dominated learners from other learners of lower quality. If two layers of learners are maintained however, we can maintain the tests that *separate* these layers from one another, i.e. tests that show the dominance of learners in the first layer over those in the second layer.

The maintenance of the first two layers and the tests that separate them results in an archive that is stable under repeated application of the update procedure, as long as no new learners or tests arrive. This can be seen as follows: if tests that separate the two layers of learners are maintained, then the next update step for the learner archive will leave the archive unchanged, as it again selects the first two layers of learners. Likewise, the second iteration of the test update procedure will again result in the same set of tests.

A two-layer archive is not stable in general however; when new individuals arrive, the stability of the archive can be violated. Figure 3 provides an example of this: first, three layers of learners ($L_1 - L_3$) are present. Once the third layer (L_1) is removed, test T_1 becomes superfluous. Next, when learner L_4 arrives, learners L_2 through L_4 form three layers, analogous to the initial situation (the outcomes for T_2 and T_3 are the same as those of the initial learners for T_1 and T_2). Thus, the third learner layer (L_2) and test T_2 are removed. The same principle is repeated after the arrival of L_5, now resulting in the removal of L_3, the last learner known to solve T_1.

3) Three layers: Given the knowledge that two layers are insufficient to avoid regress, we may consider to what extent adding additional layers may increase the reliability of a layered archive. When three layers are maintained, four layers must be present before a learner can be removed. The construction of the example is analogous to the previous examples, and is shown in figure 4.

	T1	T2	T3	T4	T5	T6
L1	0	0	0			
L2	1	0	0	0		
L3	1	1	0	0	0	
L4	1	1	1	0	0	0
L5		1	1	1	0	0
L6			1	1	1	0
L7				1	1	1

Fig. 4. Three-layer example leading to regress.

Since three learners are required to have four layers and removing these initial learners requires the addition of three more learners, it appears that the example in figure 4 is minimal, though we have not proven this.

A comparison of the examples shows that an increasing number of specific learners and tests must be selected in order to identify a case where reliability is not provided. Indeed, the examples represent worst-case scenarios; as soon as one of the learners solving an early test also solves a later test, e.g. if L_2 would solve T_4, the learners and tests would be preserved. These observations suggest that while counter-examples can be constructed, the maintenance of a small number of layers may be sufficient for reliable progress in practice, even if reliability cannot be truly guaranteed. The layered method that has been investigated may therefore be of practical use, and will be called the LAyered Pareto-Coevolution Archive (LAPCA). Below, we summarize the algorithm and investigate its performance in experiments.

V. THE LAYERED PARETO-COEVOLUTION ARCHIVE

The LAyered Pareto-Coevolution Archive (LAPCA) is a variant of IPCA that maintains the first n layers of learners as the learner archive, and the tests that separate these layers as a test archive. The algorithm is based on IPCA, with two modifications: rather than maintaining non-dominated learners only, the first n layers of learners are maintained during archive updates. As in IPCA, only learners satisfying the $useful()$ condition are accepted into the archive. The test archive in LAPCA is updated as follows. For any two subsequent layers of learners, a separation set is determined and selected to form part of the updated version of the archive. In addition, for any learner in the learner archive, if the test archive contains a test that fails it, such a test will be part of the updated test archive. In determining which tests are to be maintained, tests are visited in a fixed order, and the first test to satisfy the selection criterion is selected. Thus, if an already selected test provides multiple functions, adding an

unnecessary additional test is avoided. The following section presents experimental results with LAPCA.

VI. Experimental Setup

A. The COMPARE-ON-ONE problem

The test problem that will be employed is a discretized version of the COMPARE-ON-ONE problem described in [22], [6]. The COMPARE-ON-ONE problem is a variant of the Numbers Game [8]. In the Numbers Games, learners and tests have equal representations; both are n-dimensional real-valued vectors. The goal for learners is to maximize their outcomes on the tests.

In COMPARE-ON-ONE, the learner and the test are compared based on only *one* of the test's dimensions, namely the dimension with the highest value. The outcome of the test is positive (1) if and only if the learner's value in the test's highest dimension is at least as high as that of the test:

$$m = \arg\max_i T_i \tag{1}$$

$$\text{compare} - \text{on} - \text{one}: \tag{2}$$

$$G_{one}(L,T) = \begin{cases} 1 & \text{if} \quad L_m \geq T_m \\ -1 & \text{otherwise} \end{cases} \tag{3}$$

where L is a learner, T is a test, and x_i denotes the value of individual x in dimension i.

The underlying objectives of COMPARE-ON-ONE simply correspond to the n dimensions of the genotype [22], [6]; any increase in a dimension enlarges the set of tests that are solved by the learner, and vice versa. Progress on COMPARE-ON-ONE can therefore be measured simply by considering the n variable values of an individual. Since we are interested in progress on *all* underlying dimensions, the *lowest* value, rather than for example the average value, is used as a performance indicator. If this value consistently increases over time, this indicates progress is being made over all underlying objectives.

The COMPARE-ON-ONE problem is valuable as a test-problem for coevolution methods for two reasons. First, since the underlying objectives are known, progress can be measured in an objective manner. Second, the progress of learners in COMPARE-ON-ONE critically depends on the maintenance of an informative set of tests, especially for more than two dimensions and when used in combination with a mutation bias. As discussed in detail elsewhere [6], tests must be maintained in different regions of the space in order to obtain information about all dimensions of the learners. Only by maintaining such an informative collection of tests, increases and decreases can be detected for all dimensions of the learners, so that progress and regress can be distinguished.

In these experiments, the three-dimensional COMPARE-ON-ONE problem is employed, and a mutation bias of 0.025 on a mutation range of 0.25 is used, meaning a value x is mutated to $x+c$ with c chosen randomly from $-0.025, 0.225$. Due to the use of a mutation bias, the values of individuals (and thereby their evaluation) is more likely to decrease than to increase under mutation, thereby making regress likely when mutation is used. When mutation is used, two dimensions of the individual are mutated; if only one dimension were mutated, it would still be relatively easy to detect progress, as any detected improvement would constitute a true overall improvement. By mutating two dimensions at a time, it becomes substantially more difficult to determine this, as an increase or decrease in one of the dimensions may go undetected. As a result, distinguishing progress from regress requires maintaining an informative set of tests that provides accurate evaluation. The form of mutation that has been described renders the test problem difficult. It is intended to model the property of problems of practical interest that the variation operator is typically more likely to produce regress than progress. It is used to produce 50% of all offspring; the other half is produced using conventional two-point crossover.

In previous work, several coevolution methods have been compared on the COMPARE-ON-ONE problem [22], [6]. The only algorithm found to provide stable progress was the DELPHI algorithm, and a variant also based on the use of distinctions for the evaluation of tests. A limitation of DELPHI however is that the algorithm its restricted in the use of exploration; since tests can only be replaced by their own offspring, the algorithm cannot be used as an archive to which a generator may supply diverse new individuals. The IPCA and LAPCA methods are designed to overcome this issue.

B. The Discretized COMPARE-ON-ONE problem

To test whether an archive-based approach can confer benefit for problems requiring exploration, we define a variant of COMPARE-ON-ONE for which exploration is required; this is achieved by discretizing the search space before the outcome of a learner on a test is computed.

The discretization procedure operates as follows. We define a regular grid consisting of hyper-cubes of size $\delta = 0.25$ in each dimension, starting at the origin. To discretize the individuals, we round the value in each dimension of the learner and test to the first grid-point below it before evaluation, without affecting the genotype. Thus, a learner $[0.23, 0.30, 0.47]$ would be mapped to $[0, 0.25, 0.25]$ before computing its outcome on a test, which is discretized likewise.

The discretization makes it necessary to perform exploration. Suppose there are two individuals; one existing individual with a value of 0.1 for some dimension, and another individual with a value of 0.15 for the same dimension. Now whereas in the continuous version the difference between these individuals can be detected by any test whose highest value is in the same dimension and lies between these values, for the discretized version no test can detect the benefit conferred by the new individual. Thus, the only way to progress is by performing random exploration until progress can be detected. This furthermore requires the simultaneous availability of appropriate tests. The discretization procedure thus greatly reduces the amount of gradient present in the problem.

C. Generator

Archive algorithms operate in conjunction with a *generator* of new individuals. The generator in these experiments is an

explorative coevolutionary algorithm that maintains a population of learners and a population of tests, and will now be described.

The objectives for learners are their outcomes against tests and the distinctions between tests. The learner objectives are based on the union of the current population and new generation of tests. The objectives for the tests are analogous, namely their outcomes against and distinctions between individuals in the current population and new generation of learners, resulting in a symmetric setup.

For each objective achieved by an individual, a score is assigned that equals one over the number of other individuals that achieve the objective. The idea behind this form of *competitive fitness sharing* [14] is that objectives addressed by very few individuals count more than objectives addressed by all individuals. The sum of an individual's scores on the n outcome objectives and on the n^2 distinction objectives, where n is the size of the population plus the new generation, are added to yield a single total score for the individual.

Selection is based on the scores of individuals. Given a current population and a new generation of individuals, the number of population members k to be replaced is determined adaptively. To compute k, the highest scoring individuals of the new generation are lined up with the lowest scoring individuals of the current population. Then k is determined as the highest number for which the summed scores of the first k generation members is still at least as high as that of the first k population members. Based on this procedure, the lowest scoring k population members are discarded. The population members are replaced by k *randomly* selected individuals from the new generation, thus yielding an explorative generator. Preliminary experiments suggested that using this exploratory method as a generator, reliability can be achieved at a lower number of layers than when the more greedy generator method of selecting the n highest scoring generation members is used.

Both the learner and the test population have population-size 10. Offspring is generated using crossover (50%) and mutation (50%). Mutation has been described above. When choosing a parent to generate offspring, with probability 0.1 an archive member is used as a parent. When selecting a parent from the archive, a learner is always chosen from the first layer, as these learners have the best performance obtained so far. Test archive parents are selected uniformly, as early tests may still be required to provide informative evaluation. Performance is plotted as a function of the number of generations, and is averaged over 50 runs.

VII. Experimental Results

In previous work, the performance of the DELPHI algorithm and IPCA on the COMPARE-ON-ONE problem has been compared [18]. It was found that whereas DELPHI is able to address the continuous version of the problem efficiently, it is unable to make progress on the discretized version. The IPCA algorithm makes sustained progress on both versions of the problem.

Here, we aim to see whether sustained progress on the discretized COMPARE-ON-ONE problem can be made using a size-limited archive. To this end, the LAPCA algorithm is employed using 1, 2, 5, 10, and 20 layers. Figure 5 shows the results. For very small numbers of layers (1 or 2), progress cannot be guaranteed, as expected. For 5 layers, progress is made for over 5,000 generations, after which the minimum value drops, indicating tests for some underlying objective are no longer present in the test archive. For 10 and 20 layers however, progress continues to be made up to the end of the runs. LAPCA thereby provides a size-limited algorithm that is found to be reliable on a difficult test problem.

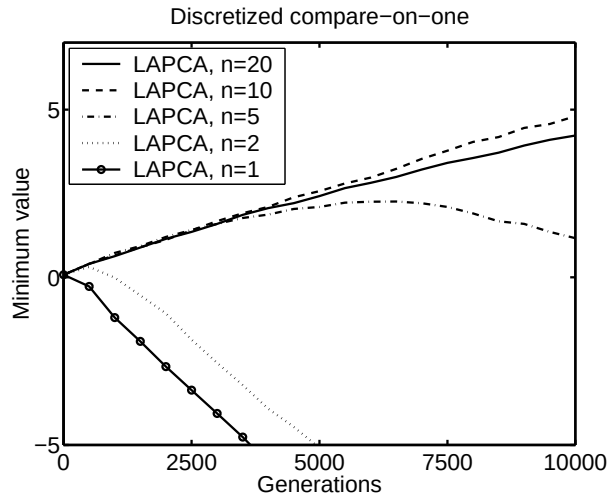

Fig. 5. Performance of LAPCA on the discretized COMPARE-ON-ONE problem.

The performance of the 10-layer archive improves at a slightly faster pace than that of the 20-layer archive. Since performance is measured in terms of the number of generations, this is not due to an increased number of fitness evaluations that must be processed due to a larger archive size. Instead, we hypothesize that the difference may be due to the selection of parents from the test archive.

Figure 6 compares the results for IPCA with the 10-layer version of LAPCA. The comparison shows that while for IPCA the rate of progress levels off over time, this is not the case for IPCA. Thus, even when performance is measured in terms of generations, and thus doesn't reflect the increasing evaluation times for IPCA due to a growing archive, LAPCA improves at a higher rate.

A central factor of interest here is the development of archive sizes over time. The size of the learner archive is of some importance as new learners must be compared on all objectives against the learner archive. The main factor of interest however is the test archive size, as new learners must be evaluated on every test in the archive.

Figure 7 shows the size of the learner and test archives for IPCA. While the learner archive is limited in size, as determined to a large extent by the test problem, the test archive grows steadily over time. This results in increasing evaluation times over the course of a run, and ultimately limits the potential for application of IPCA.

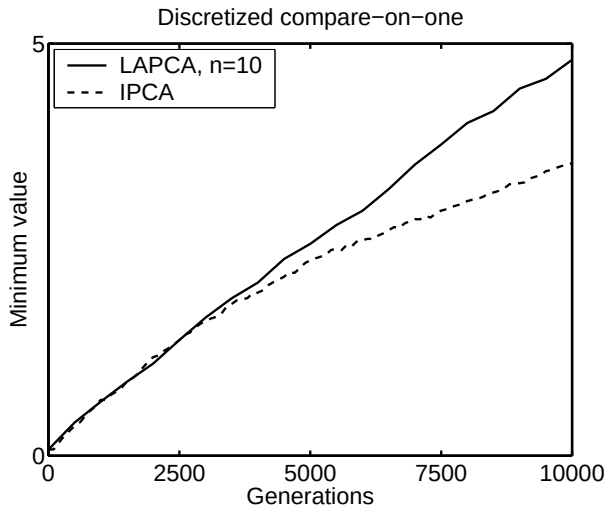

Fig. 6. Performance of IPCA and 10-layer LAPCA on the discretized COMPARE-ON-ONE problem.

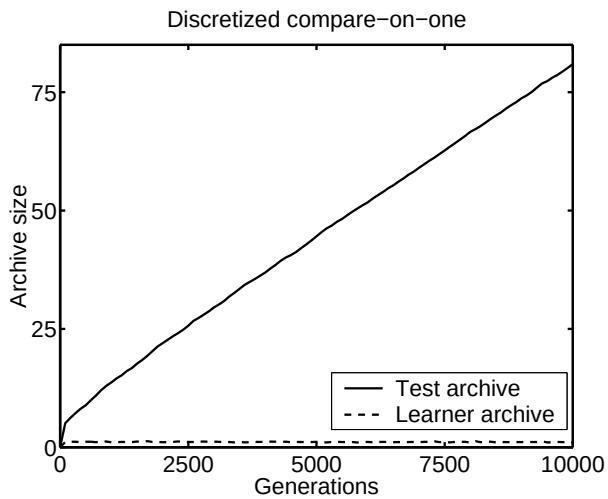

Fig. 7. Size of learner and test archives for IPCA.

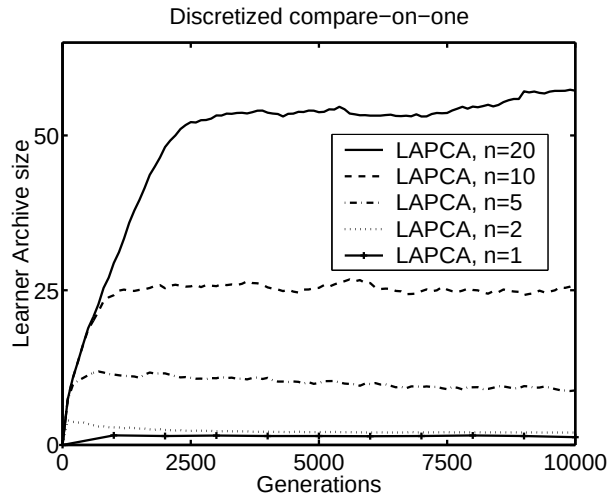

Fig. 8. Size of learner archives for LAPCA.

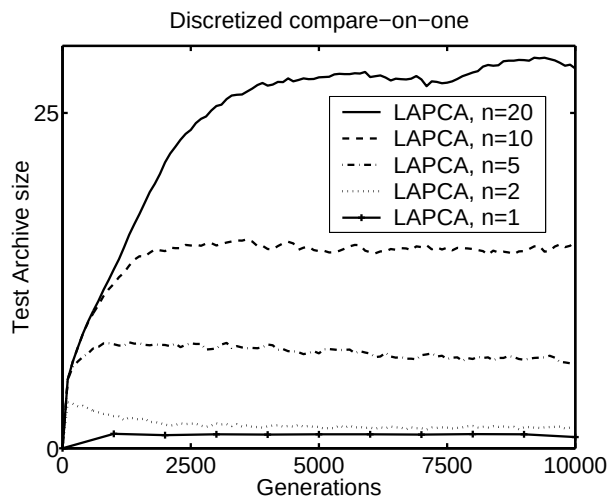

Fig. 9. Size of test archives for LAPCA.

Figures 8 and 9 show the learner and test archive sizes for LAPCA. As expected, the size of the archive increases with the number of layers. A surprising feature however is that the sizes of both the learner and test archives do not continue to grow, but level off over time. This can be explained by the fact that the Pareto-front and further layers need not be large in size; if a cluster of individuals moves along the diagonal of the space, sustained progress is possible while maintaining small layers.

A further observation is that the size of the test archives is substantially smaller than that of the learner archives and remains much smaller than for IPCA. Thus, for the problem at hand, the LAPCA algorithm is successful at reducing the archive size, while still providing stable progress.

Finally, we compare the computational cost of the different algorithms that have been investigated. Figure 10 shows the number of fitness evaluations as a function of the number of generations. While this number is closely linked to the archive sizes, it is dependent on factors such as the number of individuals that must be compared in determining dominance. The comparison shows that while the computational costs for IPCA grow at an increasing rate, the costs for LAPCA increase approximately linearly over time, and the amount of computational expense per generation is thus approximately constant.

While the results that have been presented are problem-dependent, they show that reliable progress on a difficult test problem can be achieved using limited size archives. Hopefully, the LAPCA algorithm may thus form a step towards reliable coevolution techniques for practical problems. A main limitation of LAPCA is that the size of layers may still be large. Since this is also a problem in EMOO in general, techniques used to address this issue in standard EMOO algorithms can in principle be transferred to Pareto-Coevolution.

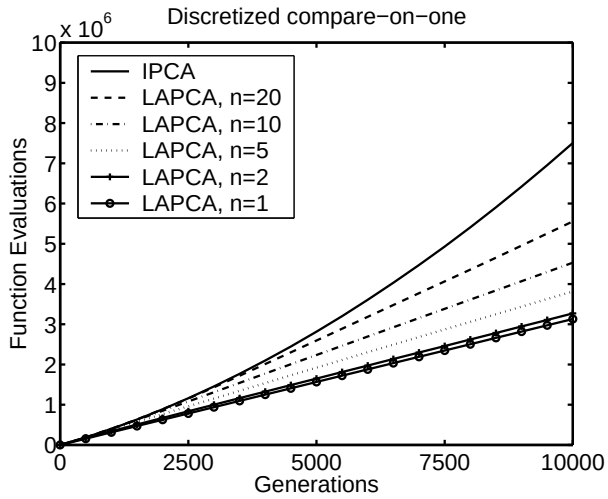

Fig. 10. Size of test archives for LAPCA.

VIII. Conclusions

Recently, archive methods have been introduced that guarantee progress in coevolution without posing overly strict conditions on the acceptance of individuals into the archive. An important question in the practical application of these archives is how the size of an archive may be limited while still providing reliability.

We investigate a variant of the Incremental Pareto-Coevolution Archive called the LAyered Pareto-Coevolution Archive (LAPCA). LAPCA maintains the first n layers of learners along with tests that separate these layers from one another. By varying the number of layers, the reliability of the archive can be tuned. Thus, reliability may be maximized given an available amount of computational resources.

The LAPCA algorithm is investigated in experiments, and found to provide reliable progress on the discretized COMPARE-ON-ONE problem which requires exploration. This is achieved using only small learner and test archives, and the archive sizes were moreover found to be approximately stable for the problem that has been investigated. While practical problems may feature much larger layers and thereby result in a larger archives, LAPCA represents the first general Pareto-Coevolution archive of tunable reliability that features a bounded size, and may thereby bring practical, reliable algorithms for coevolution a step closer.

References

[1] D. W. Hillis. Co-evolving parasites improve simulated evolution in an optimization procedure. *Physica D*, 42:228–234, 1990.

[2] Christopher D. Rosin and Richard K. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.

[3] Hugues Juillé and Jordan B. Pollack. Coevolving the "ideal" trainer: Application to the discovery of cellular automata rules. In *Proceedings of the Third Annual Genetic Programming Conference*, Madison, Wisconsin, 1998.

[4] Jan Paredis. Coevolutionary computation. *Artificial Life*, 2(4), 1996.

[5] Ludo Pagie and Paulien Hogeweg. Evolutionary consequences of coevolving targets. *Evolutionary Computation*, 5(4):401–418, 1998.

[6] Edwin D. De Jong and Jordan B. Pollack. Ideal evaluation from coevolution. *Evolutionary Computation*, 12(2), 2004.

[7] D. Cliff and G. F. Miller. Tracking the Red Queen: Measurements of adaptive progress in co-evolutionary simulations. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, editors, *Proceedings of the Third European Conference on Artificial Life: Advances in Artificial Life*, volume 929 of *LNAI*, pages 200–218, Berlin, 1995. Springer.

[8] Richard A. Watson and Jordan B. Pollack. Coevolutionary dynamics in a minimal substrate. In L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-01*, pages 702–709, San Francisco, CA, 2001. Morgan Kaufmann.

[9] Karl Sims. Evolving 3D morphology and behavior by competition. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 28–39, Cambridge, MA, 1994. The MIT Press.

[10] Hugues Juillé and Jordan B. Pollack. Co-evolving intertwined spirals. In L.J. Fogel, P.J. Angeline, and T. Baeck, editors, *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pages 461–467, Cambridge, MA, 1996. The MIT Press.

[11] Jordan B. Pollack and Alan D. Blair. Co-evolution in the successful learning of backgammon strategy. *Machine Learning*, 32(1):225–240, 1998.

[12] Faustino J. Gomez and Risto Miikkulainen. Solving non-markovian control tasks with neuro-evolution. In Thomas Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence, IJCAI-99*, pages 1356–1361, San Francisco, CA, 1999. Morgan Kaufmann.

[13] Mitchell A. Potter and Kenneth A. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.

[14] Christopher D. Rosin. *Coevolutionary Search among Adversaries*. PhD thesis, University of California, San Diego, CA, 1997.

[15] 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 July 2002. AAAI.

[16] Sevan G. Ficici and Jordan B. Pollack. A game-theoretic memory mechanism for coevolution. In E. Cantú-Paz et al., editor, *Genetic and Evolutionary Computation – GECCO-2003*, volume 2723 of *LNCS*, pages 286–297, Chicago, 12-16 July 2003. Springer-Verlag.

[17] Lothar M. Schmitt. Theory of coevolutionary genetic algorithms. In Minyi Guo and Laurence Tianruo Yang, editors, *Parallel and Distributed Processing and Applications, International Symposium, ISPA 2003*, pages 285–293, Berlin, 2003. Springer.

[18] Edwin D. De Jong. The Incremental Pareto-Coevolution Archive. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-04*, 2004.

[19] Sevan G. Ficici and Jordan B. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Julian Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature, PPSN-VI*, volume 1917 of *LNCS*, Berlin, 2000. Springer.

[20] Richard A. Watson and Jordan B. Pollack. Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Julian Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature, PPSN-VI*, volume 1917 of *LNCS*, Berlin, 2000. Springer.

[21] Sevan G. Ficici. *Solution Concepts in Coevolutionary Algorithms*. PhD thesis, Brandeis University, 2004.

[22] Edwin D. De Jong and Jordan B. Pollack. Learning the ideal evaluation function. In E. Cantú-Paz et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-03*, pages 274–285, Berlin, 2003. Springer.

[23] Anthony Bucci, Jordan B. Pollack, and Edwin D. De Jong. Automated extraction of problem structure. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-04*, 2004.

[24] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[25] Sevan G. Ficici and Jordan B. Pollack. Pareto optimality in coevolutionary learning. In Jozef Kelemen, editor, *Sixth European Conference on Artificial Life*, Berlin, 2001. Springer.