The Incremental Pareto-Coevolution Archive

Edwin D. de Jong

Decision Support Systems Group, Universiteit Utrecht
PO Box 80.089, 3508 TB Utrecht, The Netherlands
dejong@cs.uu.nl
http://www.cs.uu.nl/∼dejong

Abstract. Coevolution can in principle provide progress for problems where no accurate evaluation function is available. An important open question however is how coevolution can be set up such that progress can be ensured. Previous work has provided progress guarantees either for limited cases or using strict acceptance conditions that can result in stalling. We present a monotonically improving archive for the general asymmetric case of coevolution where learners and tests may be of distinct types, for which any detectable improvement can be accepted into the archive. The Incremental Pareto-Coevolution Archive is demonstrated in experiments.

1 Introduction

Coevolution [2, 1, 11] can be seen as an approach to the problem of fitness function design. Using coevolution, the evaluation function itself can be adapted as part of the evolutionary process. This approach can be of value if the quality of individuals can be assessed using some form of *tests*, and evaluating individuals on all tests is infeasible. For such *test-based problems*, the identification of an informative set of tests can reduce the amount of required computation, while potentially providing more useful information than any static selection of tests. Several previous authors have explicitly distinguished between the role of optimizing performance and the role of assessing the performance of other individuals [11, 14, 13, 12, 9] while a main other view of coevolution is that in which coevolving individuals form components of a whole [15, 21, 19]. Here, individuals used for testing will be called *tests*, while individuals whose performance we wish to optimize are called *learners*.

A main question in coevolution is how progress may be guaranteed. Before the notion of progress in a coevolutionary algorithm can be considered, a *solution concept* [10] must be chosen. An example of a solution concept would be the learner that solves the largest number of tests. However, there may well be learners that solve tests not solved by such a learner, and may therefore be valuable. Furthermore, the specific outcomes of a learner against tests represent valuable information, and can be helpful in exploring a diverse set of learners.

A solution concept that employs all information about learners provided by tests is given by Pareto-Coevolution. In Pareto-Coevolution [8, 20], tests are

treated as *objectives* in the sense of Evolutionary Multi-Objective Optimization. The resulting solution concept is the Pareto-front, containing all learners that are non-dominated as determined by their test outcomes.

Broadly, there are two approaches to the aim of guaranteed progress in coevolution. The first approach is to strive towards accurate evaluation; if this can be achieved, then progress can be guaranteed simply by using an elitist selection mechanism based on the coevolutionary evaluation function. The second is to maintain an archive whose quality increases monotonically according to some performance criterion. The first approach is taken e.g. in [7] with the DELPHI algorithm; this approach will be discussed briefly below. Here, we will be concerned with the second, archive-based approach.

1.1 Reliable Progress by means of Accurate Evaluation

One approach to reliable progress in coevolution is to consider how tests can be evolved that provide accurate evaluation. Several authors have investigated the accuracy of coevolutionary evaluation [16, 12, 9, 3]. Based on Ficici's notion of *distinctions* [9], it has been shown that coevolution can in principle provide ideal evaluation [6, 7]. The DELPHI algorithm is based on this principle, and will be used in comparison experiments here. For a discussion of the algorithm, its motivation, and experimental results, the reader is referred to [7].

1.2 Archive-based Methods for Monotonic Improvement

A common technique in coevolution aimed at improving reliability is the use of an archive. Several archive mechanisms exist that are intended to improve the reliability of coevolutionary algorithms but do not provide any specific guarantees. Here, since our aim is to study how progress may be guaranteed, we will be concerned solely with archives that provide some form of progress guarantee.

The process that supplies new individuals to the archive will be referred to as the *generator*. The generator will typically use the archive for testing purposes, but an archive may also be valuable as a basis for generating new individuals. Any progress guarantee relies on the ability of the generator to produce new individuals. The aim for an archive is therefore to guarantee that regress is avoided; if this is guaranteed, then any changes in performance must represent progress in some aspect, as will be made precise.

A central requirement for a coevolution archive is that is should guarantee monotonic progress. Apart from this requirement, there are at least three other characteristics that determine the practical value of the archive:

- **Generality** *Generality* reflects the scope of the archive method; an archive that guarantees progress for all forms of coevolution would be maximally general.
- **Sensitivity** An archive is *sensitive* if it is able to detect small improvements in the quality of learners. This property applies to both learners and tests. If

an archive is sensitive in accepting learners, it can accept many of the learners that represent improvement, which positively affects both the generation of new learners and the evaluation of future tests. If an archive is sensitive in accepting tests, it will subsequently be more likely to detect improvements made by learners. The property of sensitivity necessarily depends on the solution concept.

– **Efficiency** An archive is *efficient* if it consumes a limited amount of resources, notably computation time and storage capacity.

The majority of archives employed in the coevolution literature can be described as *best-of-generation* models, where the archive contains the fittest learners of the m past generations, and a sample of the archive is used for testing the current learners [10]. In such setups, tests are selected based on their quality as learners, rather than on their ability to provide informative evaluation. The maintenance of individuals performing well against a sample of previous learners is not by itself sufficient to guarantee progress in coevolution. In the following, we will discuss methods that do provide a progress guarantee.

Rosin describes the *covering competitive algorithm* [16], which alternates between finding a first-player strategy that beats all second-player strategies in the archive and *vice versa*. Under the assumption of an unbounded archive, the covering competitive algorithm guarantees monotonic progress. The algorithm assumes the existence of a first-player strategy that defeats *all* second-player strategies. For many test-based problems however, no learner can simultaneously achieve the highest attainable score on all possible tests, as there can be trade-offs between the different tests.

If the covering competitive algorithm is to be used for a problem featuring multiple underlying objectives and thus possibly more than one Pareto-optimal learner, every such learner on the Pareto-front would form a local optimum; whenever the method finds one learner on the Pareto-front and the tests it solves, no further progress can be made, and the method will stall. A similar argument holds for the *dominance tournament* [18], which was proposed as a method for tracking progress in coevolution but can also be used as a coevolutionary archive [10].

Schmitt [17] presents a stochastic model intended to demonstrate that coevolution can converge to a global optimum if for at least some species strictly dominant individuals exist that maximize performance over all possible evaluation environments. Here, the aim will be to guarantee progress under broader, less strict conditions.

A recent archive mechanism providing a progress guarantee is the *Nash memory* [10], which employs the Nash equilibrium as a solution concept. A mixed strategy Nash equilibrium is a combination of mixed strategies such that no player can profitably deviate given the strategies of the other players. An attractive feature of the Nash equilibrium as a solution concept is that the set of learners it represents can be relatively small compared to the Pareto-front, which is a valuable property for coevolutionary search. A disadvantage is that

there can be many Nash equilibria, part of which may be dominated; thus, a Nash equilibrium does not necessarily achieve the highest outcomes possible.

Given an unbounded memory, the Nash-memory archive can provide monotonic progress for symmetric games, i.e. problems where learners and tests come from the same space. The Nash memory consists of two sets, $\mathcal{N}$ and $\mathcal{M}$. The goal for $\mathcal{N}$ is to approximate the solution concept, and to improve monotonically over time in doing so. If $\mathcal{M}$ is of bounded size, arbitrary strategies may be pruned from it, and monotonic progress of $\mathcal{N}$ is not guaranteed. If $\mathcal{M}$ is of unbounded size however, monotonic progress of $\mathcal{N}$ can be guaranteed.

The solution concept employed in this work is that provided by Pareto-Coevolution, and consists of the Pareto-optimal set that results from using tests as objectives. This set is the set of all learners than cannot improve their performance on any test without lowering their performance on some other test.

We will consider how monotonic improvement can be achieved for unrestricted Pareto-Coevolution. Thus, learners and tests are allowed to be of different types, and the Pareto-optimal set is employed as a solution concept. We present an algorithm called the Incremental Pareto-Coevolution Archive (IPCA), and prove that it guarantees monotonic progress. The IPCA is demonstrated in experiments.

The paper is structured as follows. Section 2 defines the notion of monotonic progress, and Section 4 discusses some first possibilities for achieving monotonic progress in Pareto-Coevolution. In Section 4, we present the IPCA algorithm, which guarantees monotonic progress for Pareto-Coevolution. Experimental results are described in Section 5, followed by conclusions.

2 Defining Monotonic Progress

To determine whether progress is monotonic, we must be able to compare different approximations of the solution concept. We must therefore specify which property of a Pareto-front should improve over time. In this paper, we will assume tests are binary. We will say a learner *solves* a test set if it obtains a positive score for every test in the set. Thus, the relevant performance criterion in Pareto-Coevolution is which sets of tests can be solved by some single learner. If the collection of test sets that can be solved by the learner archive grows monotonically over time, then monotonic progress is guaranteed.

To determine whether the collection of test sets solved by single learners grows monotonically, two requirements must be satisfied. First, the collection may not shrink; if a test set can be solved by a learner at time t, then the test set must be solved by some learner at any time t' after t. This guarantees that regress is avoided. Second, to ensure actual progress, any transition from a learner archive to its successor must increase the collection of test sets solved. Thus, the successor of the learner archive must solve some test set that is not solved by the current archive.

The learner archives obtained over time form a series of approximations of the solution concept. These approximations will be denoted as $L^1, L^2, \ldots L^t$, while

the test archives obtained over time will be written as $T^1, T^2, \ldots T^t$. Formally, the above requirements can now be stated as follows:

Definition 1 (Monotonic Progress). *Let $L^t, L^{t'}, T^t$, and $T^{t'}$ be sets of learners and tests at times t and t'. Then monotonic progress is achieved if for any $t, t' > t$:*

$$\begin{aligned} 1.\ \forall TS \subseteq T^t : \quad & [\exists L \in L^t : solves(L, TS) \implies \\ & \quad \exists L' \in L^{t'} : solves(L', TS)] \\ 2.\ \exists TS \subseteq T^{t'} : \quad & [\nexists L \in L^t : solves(L, TS) \ \wedge \\ & \quad \exists L' \in L^{t'} : solves(L', TS)] \end{aligned}$$

First, progress can be guaranteed by keeping all learners and tests. This will generally be too costly however; a main question therefore is which individuals may be discarded while still retaining the guarantee of monotonic progress.

3 Monotonic Progress for Pareto-Coevolution

Now that monotonic progress has been defined, we can ask how it may be achieved by coevolutionary algorithms. A first possibility that may spring to mind is to maintain the Pareto fronts of learner and test populations, using the outcomes as objectives. An example shows that this strategy does not guarantee progress. Suppose a learner solves test A but not B. Then removing dominated tests results in loss of A. Next, a learner arrives that solves B but not A. Since A is no longer present, this learner appears to dominate the first learner and will thus replace it, so that the capacity to solve test A is lost.

The reason why a valuable learner can not be maintained in the above example, is that the tests do not provide a stable basis for evaluation; if both tests are maintained, the two learners are both non-dominated. This observation suggests that it may be useful to maintain multiple layers of learners and tests. Suppose that in addition to the non-dominated learners, we maintain all learners that become non-dominated when the non-dominated learners are removed. Furthermore, to distinguish non-dominated learners from the learners in this second layer, we maintain all tests that make a *distinction* [9] between learners from layer 1 and 2. A test makes a distinction between learners A and B if it assigns a higher outcome to A than to B. Finally, we maintain tests that cause the outcome vectors of two learners in a layer to be different, to prevent such learners from appearing identical. This setup can be generalized to maintain n layers of learners and the tests that separate them.

An interesting question is whether there is any number of layers n that is sufficient to avoid regress. This question can be answered negatively; for any n, a counter-example can be constructed as follows. First take n tests and $n + 1$

learners, and assign the outcomes such that each learner i is dominated by its successor $i+1$; this can be done by letting each learner solve all tests solved by its antecessor plus one extra test. The learner update procedure will now result in the removal of the first learner, as it resides in layer $n+1 > n$. All of the remaining learners solve the test solved by learner 1, making this test superfluous. Next, repeat the following procedure n times: add a new learner that solves all tests solved by the existing learners, and a new test solved only by the newly added learner. After each addition, the update procedure causes the most ancient learner and test to be removed from the archive. After n cycles, all learners solving the first test will have been discarded. Since the remaining learners were added after the first test was removed and hence do not solve it, the ability to solve the first test has been lost, and regress has thus occurred. This proof sketch demonstrates that no number of layers n is sufficient to guarantee the avoidance of regress.

4 The Incremental Pareto-Coevolution Archive (IPCA)

In this section, we describe an archive-based algorithm that guarantees monotonic progress for Pareto-Coevolution without simply keeping all learners or tests. The algorithm is called the *Incremental Pareto-Coevolution Archive* (IPCA). 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:

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

where $G(L,T)$ is the outcome of learner L against test T, and $\succ$ represents Pareto-dominance. 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 determine 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 some or all of $T_1, T_2, \ldots T_k$ are useful tests and a subset of TG with this same 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 follows as follows.

```
L^0 := ∅
T^0 := ∅
t := 0
while ¬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} ∪ TS
  for i = 1 : |LG|
    if useful(L_i, L^{t+1}, T^{t+1})
      L^{t+1} := L^{t+1} ∪ L_i
  end
  if L^{t+1} ≠ L^t
    t := t + 1
end
```

Fig. 1. The Incremental Pareto-Coevolution Archive (IPCA). Monotonic progress can be guaranteed for this archive-based Pareto-Coevolution algorithm.

4.1 Monotonicity and Convergence

The above algorithm is called the Incremental Pareto-Coevolution Archive (IPCA). 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. This can be guaranteed for example by generating every possible individual with a non-zero probability.

A proof that the algorithm guarantees monotonic progress as defined by Definition 1 is provided in the Appendix. 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.

5 Experimental Results

To demonstrate the operation of the IPCA, we now investigate its performance on test problem that requires exploration, and compare performance with the DELPHI algorithm.

In COMPARE-ON-ONE [7], learners and tests are n-dimensional real-valued vectors. A tests assesses a learner on the dimension in which the test itself is highest. It assigns a score of 1 if the learner is at least as high as the test in this dimension, and -1 otherwise.

In the discretized COMPARE-ON-ONE problem, the value in each dimension of the learner and test is rounded to the nearest multiple of $\delta = 0.25$ below it before evaluation, without affecting the genotype. Thus, $[0.23, 0.30, 0.47]$ is mapped to $[0, 0.25, 0.25]$. This procedure greatly reduces the amount of gradient present.

To further increase the difficulty of the test problem, a mutation bias of 0.025 on a mutation range of 0.25 is used, meaning mutation adds a value randomly chosen from $[-0.15, 0.1]$. This bias towards regress is intended to model the situation in problems of practical interest, where the variation operator is typically more likely to produce regress than progress.

The generator that supplies candidate learners and tests to IPCA produces offspring using crossover (50%) and mutation (50%). With probability 0.1, it uses an archive member as a parent. The generator maintains a learner and test populations, both of size 10. 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, as in *competitive fitness sharing* [16]. 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.

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. The lowest scoring k population members are discarded and replaced by k *randomly* selected individuals from the new generation, thus yielding an explorative generator.

The performance criterion is the lowest value among all dimensions of an individual; if this value increases, progress is made on all dimensions. Performance is plotted as a function of the number of actual generations, and averaged over 50 runs.

Figure 2 shows the behavior of the DELPHI algorithm on both the standard and discretized COMPARE-ON-ONE problem with mutation bias. While DELPHI achieves stable progress on the standard COMPARE-ON-ONE problem, it fails on the discretized version of the problem. This is expected given the operation of the method; since new individuals can only be accepted into the population if they dominate an existing individual, the method cannot make progress on problems where exploration is required before such improvements can be identified.

Figure 2 shows the behavior of the IPCA on the same two problems. While IPCA improves slower than DELPHI on the continuous problem, it does make reliable progress, as expected. Moreover, IPCA makes substantial and reliable progress on the discretized problem as well. The main limitation of IPCA is

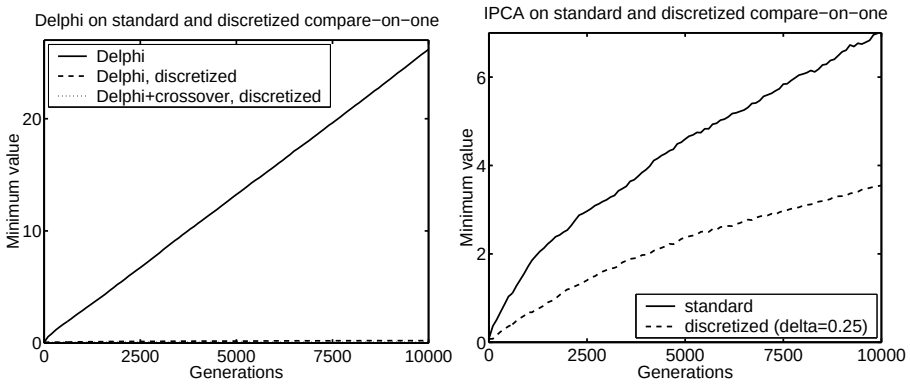

Fig. 2. Left: Performance of the DELPHI algorithm on the standard and discretized COMPARE-ON-ONE problem. DELPHI works well on the standard version but fails on the discretized version, also when using crossover in 50% of the cases. Right: The Incremental Pareto-Coevolution Archive (IPCA) makes consistent progress on both versions of the problem by virtue of its monotonic progress guarantee.

the size of the test archive; as Figure 3 (left) shows, the learner archive, which consists of the current approximation of the Pareto-front, is stable and small in size, while the test archive grows steadily over time as no individuals are pruned.

As a control experiment we apply two more standard coevolution algorithms to the problem. The first method is the generator used with IPCA without the archive itself. The second is a symmetric competition coevolutionary algorithm where the tests use their outcomes against the learners as objectives, *vice versa*. Both methods result in quick regress rather than progress; see Figure 3 (right). Apparently, the methods are insufficiently selective to cope with the mutation bias, which makes regress likely unless the replacement of individuals is highly selective and based on an informative set of tests.

An interesting question is whether and how the test archive may be pruned while retaining reliable progress. In a follow-up paper, we investigate a layered variant of IPCA called LAPCA [5]. While a layered approach cannot guarantee monotonic progress, as discussed in section 3, the method can produce sustained progress on the discretized COMPARE-ON-ONE PROBLEM with small and stable learner and test archives. This method may be of some practical interest, but the question of how archive sizes may be limited at a minimal reduction of reliability remains an important open issue. The analysis of the underlying objectives or structure of a test-based problem [7, 4] may bring insight into this matter.

6 Conclusions

The Incremental Pareto-Coevolution Archive (IPCA) has been presented. The archive consists of a learner archive maintaining non-dominated individuals, in

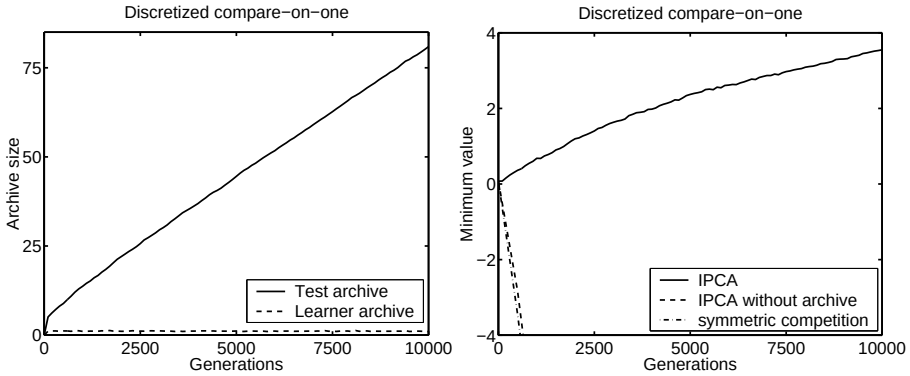

Fig. 3. Left: Sizes of the learner and test archives for IPCA. Since no tests are pruned, the test archive grows over time. Right: Comparison of IPCA with two control methods (see text). Due to the mutation bias, unreliable coevolution methods can regress, and only reliable methods can make sustained progress on the problem.

combination with an incrementally informative test archive. IPCA guarantees monotonic progress for Pareto-Coevolution.

IPCA is both general, in that asymmetric problems involving learners and tests can be addressed, and sensitive, as any nondominated learner and any test revealing new qualities of learners are accepted into the archive.

We have presented experiments based on the discretized three-dimensional COMPARE-ON-ONE problem with mutation bias. This problem requires exploration, and its mutation operator is biased towards regress. The Incremental Pareto-Coevolution Archive was found to produce sustained progress on this challenging test problem. An important remaining open question is how archive sizes may be limited at a minimal reduction of reliability.

Acknowledgements

The author wishes to thank the reviewers for detailed and thoughtful comments, and the Decision Support Systems Group at Utrecht University for a pleasant and fruitful research environment.

Appendix: Proof of Monotonic Progress

In the following, we prove that the Incremental Pareto-Coevolution Archive (IPCA) algorithm described in Section 4 guarantees monotonic progress as defined by Definition 1. The definition specifies two requirements for any t and

$t' > t$:

$$\begin{aligned} &1.\ \forall TS \subseteq T^t : \quad [\exists L \in L^t : solves(L, TS) \implies \\ &\qquad\qquad\qquad\qquad \exists L' \in L^{t'} : solves(L', TS)] \\ &2.\ \ \exists TS \subseteq T^{t'} : \quad [\nexists L \in L^t : solves(L, TS) \ \wedge \\ &\qquad\qquad\qquad\qquad \exists L' \in L^{t'} : solves(L', TS)] \end{aligned}$$

To show that monotonic progress is made over time, it is sufficient to prove that monotonic progress is made from one time-step to the next, i.e. $t' = t + 1$. This will now be shown.

Ad 1). We must show that given an $L \in L^t$ that solves TS, there must be some $L' \in L^{t+1}$ that solves TS. We distinguish between two cases: (A) L is retained, or (B) L is removed from the archive. In the first case, the requirement is satisfied by L itself. In the second case, the only situation in which a learner can be removed from the archive is if it is dominated by another learner. Let us denote this latter learner by L'. Then:

$$\forall T \in T^{t+1} : G(L', T) \geq G(L, T)$$

Since $TS \subseteq T^t \subseteq T^{t+1}$, this shows that the requirement also holds for the second case.

Ad 2). The algorithm only makes a transition from time-step t to $t + 1$ if learners have actually been added to the archive. A learner L' is only added to the archive if it satisfies the *useful* relation. Thus, there must be some $L' \in L^{t+1}$ for which

$$\begin{aligned} \nexists L \in L^{t+1} :\ L \overset{T^{t+1}}{\succ} L' \qquad \wedge \\ \nexists L \neq L' \in L^{t+1} :\ \forall T \in T^{t+1} : G(L', T) = G(L, T) \end{aligned}$$

Let TS be the set of tests solved by L': $TS = \{T \in T^{t+1} | solves(L', T)\}$. Assume $\exists L \in L^t$ such that $\forall T \in TS : solves(L, T)$. Given the second clause of the above relation, we know that there must be some $T \in TS$ for which $G(L, T) \neq G(L', t)$. Since tests are binary and L' by definition solves all tests in TS, this implies L does not solve T. This contradicts our assumption, and therefore $\nexists L \in L^t$: $\forall T \in TS : solves(L, T)$, which completes our proof. ■

References

1. Robert Axelrod. The evolution of strategies in the iterated prisoner's dilemma. In Lawrence Davis, editor, *Genetic Algorithms and Simulated Annealing*, Research Notes in Artificial Intelligence, pages 32–41, London, 1987. Pitman Publishing.
2. Nils Aall Barricelli. Numerical testing of evolution theories. Part I: Theoretical introduction and basic tests. *Acta Biotheoretica*, 16(1–2):69–98, 1962.

3. Anthony Bucci and Jordan B. Pollack. A mathematical framework for the study of coevolution. In *Foundations of Genetic Algorithms (FOGA-2002)*, San Francisco, CA, 2003. Morgan Kaufmann.
4. 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.
5. Edwin D. De Jong. Towards a bounded Pareto-Coevolution archive. In *Proceedings of the Congress on Evolutionary Computation, CEC-04*, 2004.
6. 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.
7. Edwin D. De Jong and Jordan B. Pollack. Ideal evaluation from coevolution. *Evolutionary Computation*, 12(2), 2004.
8. Sevan G. Ficici and Jordan B. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In M. Schoenauer et al., editor, *Parallel Problem Solving from Nature, PPSN-VI*, volume 1917 of *LNCS*, Berlin, 2000. Springer.
9. 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.
10. 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.
11. D. W. Hillis. Co-evolving parasites improve simulated evolution in an optimization procedure. *Physica D*, 42:228–234, 1990.
12. Hugues Juillé. *Methods for Statistical Inference: Extending the Evolutionary Computation Paradigm*. PhD thesis, Brandeis University, 1999.
13. Ludo Pagie and Paulien Hogeweg. Evolutionary consequences of coevolving targets. *Evolutionary Computation*, 5(4):401–418, 1998.
14. Jan Paredis. Coevolutionary computation. *Artificial Life*, 2(4), 1996.
15. Mitchell A. Potter and Kenneth A. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.
16. Christopher D. Rosin. *Coevolutionary Search among Adversaries*. PhD thesis, University of California, San Diego, CA, 1997.
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. 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.
19. Richard A. Watson. *Compositional Evolution: Interdisciplinary Investigations in Evolvability, Modularity, and Symbiosis*. PhD thesis, Brandeis University, 2002.
20. Richard A. Watson and Jordan B. Pollack. Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In M. Schoenauer et al., editor, *Parallel Problem Solving from Nature, PPSN-VI*, volume 1917 of *LNCS*, Berlin, 2000. Springer.
21. R. Paul Wiegand. *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, George Mason University, Fairfax, Virginia, 2003.