# A Monotonic Archive for Pareto-Coevolution

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

**Abstract**

Coevolution has already produced promising results, but its dynamic evaluation can lead to a variety of problems that prevent most algorithms from progressing monotonically. An important open question therefore is how progress towards a chosen solution concept can be achieved. A general solution concept for coevolution is obtained by viewing opponents or tests as objectives. In this setup known as Pareto-coevolution, the desired solution is the Pareto-optimal set. We present an archive that guarantees monotonicity for this solution concept. The algorithm is called the Incremental Pareto-Coevolution Archive (IPCA), and is based on Evolutionary Multi-Objective Optimization (EMOO). By virtue of its monotonicity, IPCA avoids regress even when combined with a highly explorative generator. This capacity is demonstrated on a challenging test problem requiring both exploration and reliability. IPCA maintains a highly specific selection of tests, but the size of the test archive nonetheless grows unboundedly. We therefore furthermore investigate how archive sizes may be limited while still providing approximate reliability. The LAyered Pareto-Coevolution Archive (LAPCA) maintains a limited number of layers of candidate solutions and tests, and thereby permits a trade-off between archive size and reliability. The algorithm is compared in experiments, and found to be more efficient than IPCA. The work demonstrates how the approximation of a monotonic algorithm can lead to algorithms that are sufficiently reliable in practice while offering better efficiency.

## 1 Introduction

Evolutionary Computation offers a variety of heuristic optimization methods (De Jong, 2006; Holland, 1975; Goldberg, 1989; Mitchell, 1996; Goldberg, 2002; Fogel, Owens, & Walsh, 1966; Rechenberg, 1994). Typically, these methods assume the availability of a *fitness function* expressing the relative quality of candidate solutions. For many problems of interest however, the quality of candidate solutions is determined by the outcomes of a large or infinite set of tests, and an accurate fitness function cannot be readily defined or calculated. Example domains of such *test-based problems* include learning in games, concept learning, and function approximation. The definition of an accurate and efficiently computable fitness function for a test-based problem can be a difficult problem in itself.

Coevolution (Barricelli, 1962, 1963; Axelrod, 1987; Miller, 1989, 1996; Hillis, 1990; Koza, 1992; Lindgren, 1992; Kauffman & Johnsen, 1992) avoids defining a fixed fitness function and instead identifies the tests to be used for evaluation *adaptively*, as part of the search process. By letting the search process perform its own evaluation, the biases of a fixed, hand-designed fitness function can in principle be avoided. The potential of the coevolutionary approach is so far indicated by a number of promising results. Examples include the evolution of 16-bit sorting networks (Hillis, 1990), where the coevolution of test cases (sequences to be sorted) improved the size of the smallest

 

correct network from 65 to 61 nodes; the Majority Function cellular automaton problem (Juillé & Pollack, 1998), where coevolution outperforms all other methods applied so far; the intertwined spirals benchmark problem (Juillé & Pollack, 1996), where a composite function was identified that generalizes the spiral-shaped target function outside the domain of the training data; and the evolution of complex behavior (Sims, 1994), where the adaptive evaluation of coevolution facilitates an open-ended evolutionary process.

Inspired by results such as these, a number of researchers have investigated the use of coevolution as a problem solving technique (Reynolds, 1994; Miller & Cliff, 1994; Angeline & Pollack, 1994; Ficici & Pollack, 1998; Schmidhuber, 1999; Pagie & Hogeweg, 1998; Paredis, 1996; Pollack & Blair, 1998; Funes, 2001; Rosin, 1997; Lubberts & Miikkulainen, 2001; Werfel, Mitchell, & Crutchfield, 2000; Moriarty & Miikkulainen, 1998; Potter & De Jong, 2000; Watson & Pollack, 2003; Stanley & Miikkulainen, 2004; Cartlidge & Bullock, 2004). While we focus on the application of coevolution to test-based problems here, coevolution can also be used to address problems where a fitness function is given; this form of coevolution is called *Cooperative* or *Compositional Coevolution* (Potter & De Jong, 2000; Watson & Pollack, 2003; Wiegand, 2003; Jansen & Wiegand, 2003, 2004; Bucci & Pollack, 2005)

A long-standing problem in coevolution has been that due to the adaptive nature of the test set, most existing coevolutionary algorithms are not guaranteed to make progress or to avoid regress. Thus, for most coevolutionary algorithms, spending more computational effort does not necessarily result in improvements over time.

As an example of this problem, we may imagine a population of solutions, e.g. sorting networks, that coevolves against a population of test sequences. We will say a test is *general* if the ability to solve it implies the ability to solve many other tests, and *specific* if solutions exist that solve the test and few or no other tests. Suppose that the test population becomes increasingly specific over time. Thus, a sorting network that would correctly sort all tests in the first generation would be guaranteed to sort many other sequences as well (these sequences would require a network to perform comparisons between multiple sequence elements). A network sorting all tests in the final generation however may sort few or no other sequences than those sequences themselves; the sequences in this set may require only a single swap. Then a network population that solves increasing fractions of the test populations it encounters over time would appear to be increasing in quality, as measured by its score against the test population. When its performance over all possible sequences is considered however, it is clear that the quality of solutions has actually decreased over time. This phenomenon, where the objective performance of a population can decrease as a result of evolutionary selection, has been demonstrated in experiments (Watson & Pollack, 2001).

The example demonstrates an important difference between coevolution and regular evolutionary algorithms: if solutions are evaluated based on a changing set of individuals (tests), then increases in this subjective evaluation do not imply increases in the objective quality of the solution. An important question is therefore: how can progress in the *objective* quality of individuals be guaranteed?

A main difficulty in addressing this question is that the coevolutionary search process does not have access to an objective evaluation function; indeed, it has been thought that for this reason, objective progress cannot be achieved by any coevolutionary setup. This contrasts with the situation for non-coevolutionary genetic algorithms, where elitism can be used to guarantee that the best obtained fitness will not decrease.

There are several approaches to achieving objective progress. One approach is to approximate an *ideal evaluation function*, and evolve individuals according to this evaluation function; this approach is explored in (De Jong & Pollack, 2004). Another approach is to guarantee asymptotic convergence to a solution, see e.g. (Schmitt, 2003). A third approach is to guarantee that any changes to the current approximation of the solution bring the search closer to the solution, so that as a result monotonic progress is made. Given this monotonicity, putting in more computational effort guarantees that the quality of the best solution found so far can only increase. In this article, we take the latter approach to develop an algorithm that guarantees monotonic progress.

The question of how monotonic progress can be guaranteed for various forms of coevolution is a central challenge in current coevolution research. Before this question can be studied, a desired *solution concept* must be chosen (Ficici, 2004). A solution concept is independent of a search algorithm. It specifies precisely, and in an absolute manner, which candidate solutions qualify as optimal solutions and which do not. A solution concept thus divides the set of candidate solutions in two parts: solutions and non-solutions. For example, if one is interested in maximizing a candidate solution's average score against all possible opponents, then the criterion of having a maximum average score against all opponents forms an appropriate solution concept, as it specifies a division of the search space into solutions and non-solutions. Note that the average score against a current population cannot be used to define a solution concept, as this is a relative measure that depends on the current state of a search algorithm.

Given the choice of a solution concept, the question is how monotonic progress towards the solution concept can be guaranteed. If an algorithm progresses monotonically towards the chosen solution concept, then at least the practitioner knows that by spending more computational effort, the algorithm is expected to get closer to a solution; for most previous coevolution algorithms, such a guarantee does not hold.

Several main solution concepts exist for coevolution, each with their own advantages, which will be discussed. Prior to this work, a monotonic algorithm existed for the solution concept of the Nash Equilibrium only (Ficici & Pollack, 2003). Here, we present an algorithm that guarantees monotonic progress for the solution concept of the Pareto-optimal Equivalence Set.

The algorithm that will be presented is called the Incremental Pareto-Coevolution Archive (IPCA). The monotonicity guarantee for IPCA is formally proven, and the algorithm is investigated in experiments. A practical limitation of archives with a monotonicity guarantee is that the archive size may grow indefinitely. We therefore investigate how IPCA can be pruned in a directed manner, so as to achieve a tunable trade-off between archive size and reliability. The resulting method is called the LAyered Pareto-Coevolution Archive (LAPCA). Further experiments with this method examine the trade-off between archive size and reliability[1].

The remainder of this article is structured as follows. First, in Section 2, we define several main solution concepts in coevolution. In Section 3, we review existing archive methods for these solutions concepts, and other related work. Section 4 provides a definition of monotonicity in the context of coevolution. In Section 5, IPCA is presented. Results with this method are presented in Section 6. The layered variant called LAPCA and results with this method are described in Section 7. Section 8 concludes, following which, a list of symbols is provided.

[1]The IPCA and LAPCA algorithms have previously been described in conference papers (De Jong, 2004a, 2004b)

## 2 Solution Concepts

In the following, we will discuss a number of solution concepts that can be used in coevolution. Before doing so, some terms are defined. As mentioned, we will be concerned here with coevolution as applied to test-based problems. This form of coevolution is sometimes called Competitive Coevolution, to contrast it with the other main form of coevolution, Cooperative Coevolution (Potter & De Jong, 2000); however, both forms of coevolution can exhibit both cooperative and competitive aspects. In cooperative coevolution, a fitness function is available, and the aim is to coevolve components that make up a complete solution. In test-based problems on the other hand, a static fitness function for candidate solutions does not exist; rather, candidate solutions are evaluated based on interactions with one or more types of other individuals, which will be called *tests*, as they serve to test the quality of the candidate solutions.

The quality of a candidate solution in a test-based problem is determined by its outcomes against all possible tests in the problem domain. A test can be any entity that evaluates or reveals some aspect of the qualities of a candidate solution. Examples of tests include opponents in game playing and strategy learning, test sequences in sorting, examples in concept learning, and test points in function approximation.

The distinction between candidate solutions and tests does not depend on the representation or type of the individuals, but rather on their role and place in the search process. For example, suppose one wants to evolve a population of chess players by playing it against a coevolving population of opponent players. Then the players in the two populations may have the same representation, but the candidate solutions are those in the population whose performance one wants to optimize, and the players in the opponent population used to evaluate these candidate solutions are the tests. To give another example, in the work of Hillis (1990), the performance of the sorting networks is to be optimized, while the parasites serve to evaluate the sorting networks; thus, the sorting networks are the candidate solutions, while the parasites would be named tests here.

We now proceed to formally define several main solution concepts. Each solution concept is a set specifying all objects that constitute a solution; thus, any element of a solution concept represents a solution. The following notation is used. The set of all possible candidate solutions is denoted as $\mathbb{C}$, and the set of all possible tests as $\mathbb{T}$. The outcome of a test $T$ for a candidate $C$ may in principle come from any ordered set. Without loss of generality, it will be assumed to be a real number here. The *interaction function* used to determine this outcome is written as $G$ (for Game): $G(C, T) \rightarrow \mathbb{R}$.

### 2.1 S0: Simultaneous Maximization of All Outcomes

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

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

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

### 2.2 S1: Maximization of Expected Utility

The second solution concept, Maximization of Expected Utility (MEU), specifies as a solution the individuals that maximize the expected score against a randomly selected opponent:

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

where $E$ is the expectation operator and $T$ is randomly drawn from $\mathbb{T}$. This criterion is widely used, and is appropriate for many problems, e.g., identifying a good chess player. It is equivalent to maximizing the sum of an individual's outcomes over all tests, or to a uniform linear weighting of the objectives. However, it makes the assumption that all tests are of equal importance, which limits its generality. Monotonicity for this solution concept is guaranteed by the MaxSolve algorithm (De Jong, 2005).

### 2.3 S2: Nash Equilibrium

Game theory provides the solution concept of the Nash equilibrium, e.g., (Osborne & Rubinstein, 1994). A Nash equilibrium specifies a strategy for each player such that no player can profitably deviate given the strategies of the other players. This concept can be applied to the context of test-based problems by observing that the search for a high quality individual may be viewed as the selection of a good strategy by a player. Individuals (solutions and tests) are thus viewed as strategies. Of particular interest is the mixed-strategy Nash equilibrium, where strategies do not represent single individuals but probability distributions over the space of individuals.

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

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

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

### 2.4 S3: Pareto-Optimal Set

Evolutionary Multi-Objective Optimization extends common evolutionary methods by facilitating the use of multiple *objectives*. This may be viewed as the use of a fitness function that is vector-valued rather than scalar. In Pareto-Coevolution, every possible test is viewed as an objective. A central concept in Evolutionary Multi-Objective Optimization is that of Pareto-dominance. Applied to the current context, a candidate solution $C_1$ is said to *dominate*, written $\succ$, another candidate solution $C_2$ if its outcomes against the tests are all at least as high, and if additionally its outcome on at least one test is strictly higher. Formally:

$$C1 \succ C2 \quad \equiv \quad \forall T \in \mathbb{T} : G(C_1,T) \geq G(C_2,T) \quad \wedge \quad \exists T \in \mathbb{T} : G(C_1,T) > G(C_2,T).$$

The set of all individuals that are non-dominated trade off the different capabilities of candidate solutions in different ways (i.e., a non-dominated candidate solution A may fail some tests solved by another non-dominated candidate solution B, but will then solve other tests not solved by B). The solution concept S3 consists of a singleton set containing the Pareto-Optimal Set, which in turn contains all candidate solutions that are not dominated by any other solutions:

$$S3 \equiv \{ \{C \in \mathbb{C} | \nexists C' \in \mathbb{C} : C' \succ C\} \}.$$

We note that S3 is a set whose only member is another set, namely the Pareto front. This definition is chosen for consistency with the other solution concepts; each solution concept is a set whose members are solutions. The Pareto front forms a single, unique solution to a multi-objective problem.

### 2.5 S4: Pareto-Optimal Equivalence Set

For each maximal combination of tests that can be solved, the Pareto-Optimal set contains all candidate solutions that solve it, where *solving* means obtaining a positive outcome, as is formalized in Section 4. Thus, the set may contain many equivalent candidates that each solve the same combination of tests. We define a variant of the Pareto-Optimal set that does not contain such duplicate candidate solutions. This set is defined by the requirement that for each combination of tests that can be solved, it contains at least one candidate solution that solves it. Since multiple such sets may exist, we define $S4$ as the collection of all such sets:

$$S4 \equiv \{S \subseteq \mathbb{C} | \forall TS \subseteq \mathbb{T} : \quad \exists C \in \mathbb{C} : solves(C, TS) \implies \exists C' \in S : solves(C', TS)\}.$$

The Pareto-Optimal Equivalence Set can also be defined as a set that for each member of the Pareto-front contains an equivalent candidate, i.e. one having identical outcomes for all tests. This can be seen as follows. For any element $C$ of the Pareto front, let $TS$ be the set of tests it solves. Since there exists a candidate solution that solves $TS$, the above definition of $S4$ specifies that a member of $S4$ will contain a candidate solution that solves at least the same tests. Furthermore, this candidate could not solve more tests than $C$, otherwise $C$ would not be part of the Pareto-optimal set. Thus, the candidate is equivalent to $C$.

## 3 Related Work

A first development in the coevolution work of the past decade has been that the importance of adequate evaluation has become clear. The notion that a role of coevolving individuals is to provide evaluation for other individuals is already clearly present in Hillis's work (1990). Juillé (1999) describes the *ideal trainer* (Epstein, 1994) as an evaluation function that exposes individuals to gradient and presents problems of increasing difficulty. The instantiation of this ideal trainer is domain-specific.

In further work, a link with Evolutionary Multi-Objective Optimization has been made. By considering all tests in a coevolution problem to be *objectives*, a clear goal for coevolutionary evaluation is obtained. This important idea is known as Pareto-coevolution (Ficici & Pollack, 2000; Watson & Pollack, 2000), and has been used in a number of papers since its inception (Ficici & Pollack, 2001a, 2001b; Noble & Watson, 2001; Watson & Pollack, 2003; De Jong, 2003; De Jong & Pollack, 2004).

Since evaluation in coevolution depends on a changing set of other individuals, at first sight it may appear that any form of coevolutionary evaluation is bound to be relative, and therefore that the search process may always be misguided by incomplete or inaccurate evaluation. Rosin's work (1997), to be discussed below, already showed for a limited case that objective progress is possible. More recently, employing the framework of Pareto-coevolution, it has been shown that objective evaluation is in principle possible. Specifically, for any given set of candidates, a relatively small (sub-quadratic) set of tests is guaranteed to exist that provides evaluation identical to the ideal evaluation of using all possible tests as objectives (De Jong & Pollack, 2004).

The DELPHI algorithm (ibid.) is based on the above principle, and aims to approximate objective evaluation. It was found to successfully address test problems, including the COMPARE-ON-ONE problem, that could not be addressed by other coevolutionary methods. A limitation of DELPHI however is that it poses a strong restriction on the replacement of tests by newly generated tests. Specifically, tests can only be replaced by their own offspring, thereby disallowing the incorporation of tests generated by an external search process, and limiting the explorative potential of the search.

In addition to the development of Pareto-coevolution, a number of other theoretical approaches to coevolution have been taken including order theory (Bucci & Pollack, 2002, 2003a, 2003b), game theory (Ficici, 2004; Ficici, Melnik, & Pollack, 2005; Ficici & Pollack, 2000, 2003; Wiegand, 2003; Wiegand, Liles, & De Jong, 2002, 2003), and Markov models (Liekens, ten Eikelder, & Hilbers, 2003).

In the following, we discuss existing coevolutionary algorithms which guarantee monotonicity for some solution concept. In his thesis, Rosin (1997) defines a *teaching set* as a set of second-player strategies such that for any imperfect first-player strategy $f$, there is a second-player strategy $s$ that defeats $f$. This concept represents an important step, in that it specifies a set of individuals that is sufficient to guarantee monotonicity. Based on the notion of a teaching set, the *covering competitive algorithm* (ibid.) is defined. This algorithm alternates between finding a first-player strategy (candidate solution) that defeats all previous second-player strategies (tests), and finding a second-player strategy that defeats all previous first-player strategies. The covering competitive algorithm guarantees monotonicity for the $S0$ solution concept. Schmitt (2003) presents a convergence proof for a variant of $S0$ involving more than two types of individuals ("species"), but still requiring the existence of individuals that simultaneously maximize the outcome over all individuals of other types.

Since $S1$ is contained in $S4$, the monotonicity guarantee for $S4$ of the algorithm that will be presented here also holds for $S1$. The recent MaxSolve algorithm guarantees monotonicity for the solution concept $S1$ specifically (De Jong, 2005).

Ficici's *Nash Memory* (Ficici, 2004; Ficici & Pollack, 2003) guarantees monotonicity for $S2$, the Nash equilibrium. The Nash memory algorithm is limited to symmetric zero-sum games, but a generalization to asymmetric constant-sum games is noted to be straightforward. This would form a useful extension, as tests and candidate solutions in test-based problems often are of different types.

In the remainder of this article, we consider how monotonicity can be guaranteed and approximated for $S4$, the solution concept of the Pareto-optimal Equivalence Set. As described, this solution concept is a variant of the Pareto-optimal set. Since the method that will be presented is an archive for this solution concept, a question is how it relates to archive methods for Evolutionary Multi-Objective Optimization (EMOO).

As a first converging archive for EMOO, Laumanns et al. (2002) presented a bounded archive that provably converges to an approximation of the Pareto-optimal

set within a given error limit. An issue with this method however is the choice of the parameter regulating the accuracy of the approximation, $\epsilon$; this parameter, and the ranges of the different objectives, are assumed to be known. While methods that adapt $\epsilon$ are also presented, these can be over-conservative; when a fixed size bound is used for the archive, the Adaptive Grid Algorithm by Knowles et al. (2004) may be preferable. A recent archive algorithm based on a measure of the volume of an approximation of the Pareto-front, known as the Lebesgue measure, has also yielded promising results (Knowles, Corne, & Fleischer, 2003). A further relevant finding from EMOO is that no bounded archive can maintain an ideal minimum number of points (Knowles & Corne, 2004).

There is one main difference between the problems of approximating the Pareto-optimal Equivalence Set in coevolution and approximating the Pareto-optimal set in EMOO. While in EMOO the objective functions are known beforehand, the objectives in Pareto-coevolution consist of the set of all possible tests, which is typically very large; only those tests visited by the coevolutionary algorithm so far can be used as objectives. Thus, in practice the set of objectives in Pareto-coevolution is unknown, and changes over time. Since the objectives are unknown, archive methods employed in EMOO research cannot be directly applied. However, techniques used to bound EMOO archives may possibly be transferred to be applied to coevolutionary archives.

In order to guarantee monotonic convergence, the archive employed by IPCA is unbounded. An interesting question is whether based on the techniques employed by EMOO archives, a coevolutionary archive can be designed that approximates the Pareto-Optimal Equivalence Set while guaranteeing a bounded size. We expect that this is most relevant for problems where the outcomes of tests are multi-valued rather than binary, as in such problems there is potential value in partitioning the space. An issue that will have to be addressed by such an algorithm is which *tests* should be maintained. In IPCA, this issue is addressed by selecting the tests that have been required in demonstrating the non-dominance of candidate solutions introduced into the candidate archive.

## 4 Defining Monotonicity

For clarity of presentation, tests will be assumed to have binary outcomes in the following. A candidate solution $C$ is said to *solve* a test $T$ if it has a positive outcome on the test. $C$ solves a set of tests if it solves all tests in the set, and a set of candidate solutions $CS$ solves a test set if at least one of its members does so:

$$\begin{aligned} solves(C,T) &\equiv G(C,T) > 0 \\ solves(C,TS) &\equiv \forall T \in TS : solves(C,T) \\ solves(CS,TS) &\equiv \exists C \in CS : solves(C,TS). \end{aligned}$$

To determine whether a sequence of approximations to the solution concept improves monotonically over time, we employ a distance function and consider whether the distances between the approximations and the solution concept form a strictly decreasing sequence.

An approximation $S$ of a solution concept is characterized by a set of candidates $CS$ that form the actual approximation, and a set of tests $TS$ that provide information about these candidates: $S = (CS, TS)$. We will use $S_C$ to represent the candidate set of $S$, and $S_T$ to refer to the test set.

We consider a search process that receives candidates and tests, and maintains archives of these. The sequence of candidate archives obtained over time is written as $CS^1, CS^2, \ldots, CS^n$, and the sequence of test archives obtained over time as $TS^1, TS^2, \ldots, TS^n$. At each point in time, the current state $S^t$ of the search process is characterized by the current candidate archive and by the set of all tests that have formed part of the test archive at any previous point in time: $S^t = (CS^t, \bigcup_{t'=1}^{t} TS^{t'})$.

**Definition 1 (Monotonicity)** *A sequence of approximations $S^1, S^2, \ldots, S^n$ progresses monotonically towards the solution concept $Z$ as measured by a distance function $D(X, Y)$, $X, Y \subseteq \mathbb{C} \times \mathbb{T}$, if for any $t$, $1 \leq t < n$ : $D(S^{t+1}, (Z, \mathbb{T})) < D(S^t, (Z, \mathbb{T}))$.*

Here the distance function $D$ should satisfy the three conditions of a pseudo-metric:
(i) $D(X, Y) + D(Y, Z) \geq D(X, Z)$ (the triangle inequality)
(ii) $D(X, Y) = D(Y, X)$ (symmetry)
(iii) $D(X, X) = 0$.

We now consider how such a distance function may be defined for solution concept $S4$, the Pareto-optimal Equivalence Set. To this end, we employ a measure of the number of tests sets known so far to be solvable:

$$TS(S) = |\{TS \subseteq S_T \mid solves(S_C, TS)\}|.$$

To obtain a distance function for $S4$, we can count the number of test sets known to be solved by each approximation of the solution concept and calculate the absolute difference:

$$D(X, Y) = |\ |TS(X)| - |TS(Y)|\ |. \tag{1}$$

That $D$ satisfies the above conditions for a metric can be seen as follows. Each approximation $S$ maps to an integer $|TS(S)|$. If these integers are interpreted as coordinates of points on a line, then the distance $D$ between two candidate sets equals the Euclidean distance between the corresponding points; thus, $D$ satisfies the above conditions.

The above monotonicity criterion is valuable in that it provides a precise criterion for determining whether a series of approximations progresses monotonically towards a solution concept. Since it involves calculating distances to the Pareto-optimal set $Z$ however, it cannot be calculated directly. To address this, we define an operational criterion for which equivalence with the monotonicity criterion can be shown.

The *transition criterion* that will be defined measures whether the set of test sets solved grows during a transition from one approximation of the solution concept to the next:

**Definition 2 (Transition criterion)** *Let $S^t = (CS^t, \bigcup_{t'=1}^{t} TS^{t'})$ and $S^{t+1} = (CS^{t+1}, \bigcup_{t'=1}^{t+1} TS^{t'})$ be subsequent approximations of the solution concept at times $t$ and*

$t+1$. *Then the transition from* $S^t$ *to* $S^{t+1}$ *satisfies the* transition criterion *if:*

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

This provides an operational criterion for measuring monotonicity. The transition criterion is sufficient to guarantee monotonicity as defined by Def. 1; a proof of this is given in Appendix A.

In this section, criteria have been given that guarantee monotonic progress for the solution concept of the Pareto-optimal Equivalence Set. The approach has been to view the state of a search process as an approximation of the solution concept, and consider whether the distance to the solution concept decreases monotonically.

Ficici in his thesis (2004) presents a general framework incorporating the notions of solution concepts and monotonicity, and takes a different approach. There, it is assumed that the search process prefers one state of the search process over another if every game for which the latter state is a solution is a subgame of a game for which the first state is a solution. The solution concept is determined over the individuals seen so far, and it is considered whether this solution concept over part of the search space progresses monotonically. It is found that maintaining multiple candidates with identical outcomes for the tests seen so far does not lead to a monotonic search process.

While IPCA also does not include multiple candidates with identical outcomes, doing so would not jeopardize its monotonicity as it is defined here; the transition criterion only requires the presence of certain candidates but does not exclude the presence of other candidates. This points to a difference in the two concepts of monotonicity. The monotonicity concept employed by Ficici requires progress with respect to unseen tests, so that candidates that appear identical cannot be compared in the absence of further tests. Here, this issue is addressed by considering the progress of the search process with respect to the tests seen so far. It has been seen that the broader (less restrictive) transition criterion thus obtained is still sufficient to guarantee monotonicity; a proof of this was given in Appendix A. In the following, the transition criterion will be used to develop an algorithm that guarantees monotonicity.

## 5 A Monotonic Archive for Pareto-Coevolution

It has been seen that if the transition requirement can be attained by an archive, monotonicity is guaranteed. Our aim will therefore be to devise an archive mechanism that satisfies the transition requirement. When evaluating an archive, a first question will be whether it guarantees convergence, and for what solution concept. The value of an archive method can further be characterized by three factors:

- **Receptivity** An archive is maximally receptive if it accepts all individuals that contribute to bringing the search closer to the solution concept. For the solution concept of the Pareto-optimal set, this concerns all candidates that solve a previously unsolved combination of tests and all tests that assign different outcomes than previous tests.

- **Selectivity** An archive is maximally selective if it only maintains individuals that contribute to bringing the search closer to the solution concept.
- **Efficiency** A main overall evaluation criterion for an archive is the rate of progress it establishes, measured as a function of computation time. To evaluate this, a selected performance measure can be considered as a function of the number of evaluations performed.

These factors must be taken into account when designing an archive mechanism, and will be evaluated in the empirical evaluation of the proposed archive mechanisms. Related desiderata for archiving methods in EMOO are articulated in (Knowles & Corne, 2004).

### 5.1 The Incremental Pareto-Coevolution Archive: IPCA

We describe an archive mechanism that guarantees monotonicity for the solution concept of the Pareto-optimal set. The algorithm is called the Incremental Pareto-Coevolution Archive (IPCA). IPCA consists of a candidate archive and a test archive. The algorithm provides procedures to decide which newly generated candidates and tests are included into the archive. Individuals are supplied to the archives by an independent search method, called a *generator*; the archive can be combined with any coevolutionary search algorithm to turn it into a method that guarantees monotonic progress.

The basic idea of IPCA is straightforward: the archive receives sets of candidate solutions and tests, and maintains the set of candidate solutions that are non-dominated with respect to the growing set of tests maintained by the archive so far. Of multiple candidate solutions with identical outcome vectors (i.e. identical outcomes against the tests in the archive), the algorithm will maintain only one, as more individuals with the same outcomes do not contribute to the diversity of the archive and are generally not of interest as final solutions. The test archive accepts tests that are required to demonstrate the non-dominated status of the individuals in the candidate archive.

IPCA employs a function called *useful* to determine which of the candidate solutions handed to the archive should be maintained. Specifically, a newly generated candidate is useful with respect to a set of candidates CS and a set of tests TS if it is not dominated by any candidate in CS, and if there is no candidate in CS which has equal outcomes for all tests in TS, i.e. if it is non-dominated and unique:

$$
\begin{aligned}
&useful(C, CS, TS) \equiv \\
&\nexists C' \in CS : C' \overset{TS}{\succ} C \quad \wedge \\
&\nexists C' \in CS : \forall T \in TS : G(C,T) = G(C',T),
\end{aligned}
$$

where $G(C,T)$ is the outcome of candidate $C$ against test $T$, and $\succ$ is the Pareto-dominance relation. The candidate archive is periodically updated to maintain non-dominated candidates only. The implementation of the $useful$ function is detailed in Figure 2, and described later in the text.

Using this function, the IPCA algorithm can be described as follows; see pseudo-code in Fig. 1. Given current candidate and test archives, the IPCA function accepts sets of newly generated candidates and tests, and constructs new versions of the archives. First, the new versions of the archives, $CS^{t+1}$ and $TS^{t+1}$, are initialized to equal the current versions $CS^t$ and $TS^t$. Next, for each new candidate solution, the algorithm

determines whether the candidate solution is useful given the new archives and the newly generated tests. The *find-useful* function returns two values: a boolean indicating whether the candidate $C$ is useful, and a selection of tests TSsel. If the candidate solution is useful, it is added to the new candidate archive. Any existing candidates in the archive dominated by the newly arrived candidate are removed, and any newly generated tests that were required to demonstrate the non-dominated status of the candidate solution are accepted into the new test archive. If, after considering all newly generated candidates for inclusion into the archive in this manner, the new versions of the archives are different from the previous versions, meaning individuals have been added and possibly removed from the archives, then the time step is increased to reflect the fact that the archives have been updated. Due to the increment in $t$, the current archive sets $CS^t$ and $TS^t$ will from then on refer to the newly constructed sets that were written as $CS^{t+1}$ and $TS^{t+1}$.

The *useful* function, whose implementation is described in Figure 2, works as follows. The function receives a candidate solution $C$ whose usefulness is to be assessed, candidate and test archives, and a set of newly generated tests that may be inserted into the test archive if necessary. The procedure starts from the assumption that the candidate solution is useful. For all candidates $C'$ in the candidate archive, it is determined whether the outcomes of $C'$ are at least as high as those of $C$ for all tests in the test archive. If so, *found* is set to false, and the algorithm proceeds by determining whether there is any newly generated test for which this is not the case, i.e. for which $C'$s outcome is higher. If such a test is not found, then $C$ is dominated by $C'$ or equal to it, and therefore not useful. Otherwise, $C$ is not dominated by $C'$, and the first test demonstrating $C$ is not dominated by $C'$ is moved to $Tsel$, the selection of tests that will be added to the archive if $C$ is found not to be dominated by any $C'$.

We note that the procedure that has been described, i.e. the '*useful*' function, prevents candidates with duplicate outcome vectors from entering the archive; for such candidates, no tests exist for which $C$ has a higher outcome than an existing duplicate $C'$. Thus, the useful function selects candidates that are non-dominated and unique, as desired.

### 5.2 Monotonicity and Convergence

The IPCA algorithm is designed such that any change to the archive satisfies the transition criterion specified in Def. 2. It can be seen as follows that this is the case. As noted, the *useful* function returns true if the candidate $C$ is non-dominated and unique. If useful candidates exist, IPCA will include these in the next state; otherwise, the archive remains unchanged. Therefore, if the archive has changed, a useful candidate has been added. This guarantees that the second condition of the transition criterion is satisfied; a non-dominated and unique candidate must solve a set of tests that was not previously solved by the archive (if there were an earlier candidate $C'$ that solved all tests solved by $C$, this earlier candidate would either have to be equal to $C$, or to dominate it, both of which are excluded by the positive outcome of the *useful* function). It remains to be shown that the first condition of the transition criterion holds for any change to the archive. Since IPCA only removes dominated candidates from the archive, any test set $TS \subseteq S_T^t$ that was solved by an earlier version of the archive will still be solvable after the change (if it were not, then the removed candidate would solve a set of tests not solved by the current archive, and therefore be non-dominated, which violates the assumption that it was removed).

We prove that the correspondence with the transition criterion implies that the

$IPCA.submit(Cnew,\ Tnew,\ t)\{$
  $CS^{t+1} := CS^t$
  $TS^{t+1} := TS^t$
  **for** $C \in Cnew$
    $[useful,\ Tsel] = find - useful(C,\ CS^{t+1},\ TS^{t+1},\ Tnew)$
    **if** $useful$
      $CS^{t+1} := CS^{t+1} \cup \{C\}$
      $CS^{t+1} := CS^{t+1} \setminus \{C' \in C^{t+1} | C \overset{TS^{t+1}}{\succ} C'\}$
      $TS^{t+1} := TS^{t+1} \cup Tsel$
      $Tnew := Tnew \setminus Tsel$
    **end**
  **end**
  **if** $CS^{t+1} \neq CS^t$
    $t := t + 1$
  **end**
  **return** $t$
$\}$

Figure 1: The Incremental Pareto-Coevolution Archive (IPCA). Monotonicity can be guaranteed for this archive mechanism. The candidate archive accepts candidates satisfying the *useful* criterion, and discards dominated candidates. The test archive accepts tests necessary to demonstrate the usefulness of the accepted candidates.

distance to the solution concept decreases at every step of the archive; see Appendix A for the proof. Monotonicity leaves open the possibility that an archive does not change. Actual convergence to the solution concept can be guaranteed however if the following three conditions hold:

- If the algorithm has not reached the solution concept $S4$, there always exists a step that will bring the algorithm closer to the solution concept.
- If a step exists, the individuals supplied to the archive will eventually enable the archive to make a step.
- The number of steps that can be made is finite.

The first of these conditions is proven in Appendix B. The second condition can be satisfied by choosing an explorative generator; by generating every combination of candidates and tests with some nonzero probability, which can be achieved simply by generating individuals uniformly random, any combination of individuals required to make a next step will eventually be found. Considering the third condition, a finite search space implies that the number of steps that can be made is finite. In summary, the IPCA algorithm guarantees convergence for problems with a finite search space and use of a sufficiently explorative generator.

Convergence guarantees for coevolution are important, as most basic coevolution algorithms are prone to pathological behavior and do not provide the guarantee that putting in more effort will bring the search closer to the desired optimum. However, we wish to stress that they are not sufficient to guarantee practical applicability; apart from convergence, the practical value of coevolution algorithms is determined to a high

```
FIND-USEFUL(C, Cset, Tset, Tnew){
  useful = true
  Tsel := ∅
  Trest := Tnew
  for C' ∈ Cset ∧ useful
    if ∀T ∈ Tset ∪ Tsel : G(C', T) ≥ G(C, T)
      found := false
      for T ∈ Trest ∧ ¬found
        if G(C, T) > G(C', T)
          Tsel := Tsel ∪ {T}
          Trest := Trest \ {T}
          found := true
        end
      end
      if ¬found
        useful = false
      end
    end
  end
  return [useful, Tsel]
}
```

Figure 2: The *useful* function determines whether a candidate $C$ should be maintained. For any existing candidate $C'$ in the archive that has greater or equal outcomes for all current tests, it is determined whether $C$ has a higher outcome than $C'$ on some newly generated test. If this succeeds for all $C'$, $C$ is non-dominated and unique, and thus useful.

degree by their efficiency in terms of time and space requirements. These will now be evaluated in experiments.

## 6 Experimental Results with DELPHI and IPCA

In the following experiments, we examine the limitations of the DELPHI algorithm and show that IPCA addresses these.

### 6.1 The COMPARE-ON-ONE Problem

The COMPARE-ON-ONE problem was introduced in earlier work (De Jong & Pollack, 2004). Its motivation stems from the multi-dimensional view of coevolution; in realistic coevolution problems, such as strategy learning in chess or checkers, there are multiple aspects of play that together determine the quality of a player, and a good all-round player must have adequate abilities for all aspects. Even though the precise nature of these aspects is unknown so far (we may think of qualities such as strategy and tactics), it is clear that there must be multiple such aspects; otherwise it would be possible to rank all players on a line such that a higher position on the line implies better outcomes against any opponents.

The aspects referred to here may be viewed as the *underlying objectives* of a problem, see (De Jong & Pollack, 2004). In related research, it has been shown that the underlying objectives of a problem can in principle be extracted from a matrix of game outcomes (Bucci, Pollack, & De Jong, 2004). The coevolutionary algorithm in contrast only has access to the game outcomes, and thus cannot access this objective evaluation, in line with the situation in applications of coevolution.

In the COMPARE-ON-ONE problem, the underlying objectives are explicitly represented; individuals are $n$-dimensional vectors, and each dimension represents an objective, so that the experimenter knows the objective evaluation of individuals. The correspondence between the initial objectives, where each test is an objective, and these compacted objectives, where each dimension represents a single objective, have been demonstrated in previous work (De Jong & Pollack, 2004). The solution concept of the Pareto-optimal equivalence set $S4$ for this problem corresponds to the set of non-dominated candidate solutions with unique outcome vectors, and depends on the boundaries of the space. Even without specifying boundaries however, and thereby keeping the solution domain open, we can conclude that for any dimension, higher values are preferable over lower values. The performance measure will be based on this observation.

If players are evaluated by playing against coevolving individuals, there is a risk of over-specialization; the evolved players may specialize in certain aspects of a game, e.g. the use of the rooks in chess. This is in fact quite likely if the coevolving opponents are evaluated based on their game scores; given a small initial bias towards some aspects of the game, the opponents will need to demonstrate corresponding skills in order to obtain good outcomes, leading to a positive feedback loop where only certain skills of the players are improved. As a result, performance in other aspects of the game can remain mediocre.

The above phenomenon of over-specialization is difficult to study in realistic problems, as knowledge of the underlying objectives of realistic problems is lacking so far. Yet, over-specialization may be a main factor that limits the performance of coevolutionary setups. Therefore, an abstract test problem has been constructed that permits studying the ability of algorithms to overcome over-specialization. The problem is a variant of the *Numbers Game* (Watson & Pollack, 2001), which was introduced to permit

the study of pathologies that occur in the application of coevolution.

The problem, named COMPARE-ON-ONE, is defined as follows. Individuals are real-valued vectors, and the aim of the problem is to maximize all elements of the vector. The elements of the vector represent the underlying objectives of a hypothetical problem, and the value in each dimension is a measure of the quality with respect to the corresponding objective. The problem would be trivial if the search algorithm were provided with the actual values of an individual's vector. However, as in actual coevolutionary setups, the only information that is available as a basis for selection are the outcomes of interactions between individuals. As with real problems, the outcome of these interactions depends in some way on the values for the underlying objectives.

The outcome of the COMPARE-ON-ONE game is positive (1) if and only if the candidate's value in the test's highest dimension is at least as high as that of the test:

$$m = \arg\max_i T_i \tag{2}$$

$$\mathrm{compare-on-one}: \quad G(C,T) = \begin{cases} 1 & \text{if} \quad C_m \geq T_m \\ -1 & \text{otherwise} \end{cases} \tag{3}$$

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

Due to the definition of the game, a test can only evaluate a candidate on a single dimension, namely the dimension in which the test itself has its highest value. Unless special attention is paid to maintaining a diverse set of tests, it is unlikely that the test population will maintain tests for each dimension. This effect is enhanced by the tendency of tests to increase only in the dimension on which they test, thus moving away from the diagonal. This makes it unlikely that a lost dimension will later be regained. If a dimension is lost, individuals can only progress on some of the underlying dimensions, but will be likely to drift in others. Thus, unless careful selection is performed, the test problem is highly likely to lead to over-specialization.

To increase the difficulty of the problem, a negative *mutation bias* is used; mutation is more likely to decrease a value than to increase it. This feature models the property of actual problems that a variation is more likely to cause regress than progress. Unless regress is avoided for all underlying objectives of the problem, the values of individuals are expected to actually *decrease* in the lost dimensions rather than just drift. The need to detect both progress and regress is further increased by applying mutation to multiple dimensions at the same time, as will be discussed.

To detect whether over-specialization occurs, we do not use the average value of the dimensions as a measure of quality; this value could be increasing while performance in some dimensions does not improve. Rather, the *minimum* of an individual's values is used as a measure of performance. If this value continues to increase, all dimensions must be increasing. Conversely, if over-specialization occurs, the decreasing values in a dimension will at some point result in a decreasing performance for the individual, so that overspecialization can be clearly detected. We expect that overspecialization also occurs in practical applications of coevolution but often goes undetected there, since the underlying objectives of given problems are typically unknown.

The experiments will employ the 3-dimensional version of the COMPARE-ON-ONE problem. Graphs represent the average performance of either the generator population (population performance) or the archive (archive performance), as indicated. All results are averaged over 50 runs.

### 6.2 The DELPHI Algorithm: Limited Exploration

As noted in the introduction, one approach to reliable coevolution is to let an algorithm approximate the static ideal evaluation function for the chosen solution concept as part of the search. The DELPHI algorithm takes this approach. Together with a variant, DELPHI was found in earlier work to achieve sustained progress on the COMPARE-ON-ONE problem where no other comparison methods did so (De Jong & Pollack, 2004).

The DELPHI algorithm accepts a newly generated candidate if it dominates some existing candidate, and a newly generated test if it dominates its own parent. This strict test replacement criterion functions as a heuristic that favors the maintenance of existing dimensions; it avoids replacing tests by dissimilar individuals, which would jeopardize the maintenance of existing test dimensions. This way, overspecialization is avoided, resulting in simultaneous progress on all underlying objectives.

While the ability to avoid over-specialization is valuable and a required feature for reliable coevolution methods, DELPHI has at least two limitations. First, the algorithm is designed to *approximate* ideal evaluation, but since this is not assured, reliability cannot be truly guaranteed.

The second limitation is the strict replacement criterion. The algorithm implicitly assumes that new, improved individuals can in principle be reached in a single step of the operators of variation, so that from any given point an improvement will at some point be made. For certain problems however, identifying improved individuals may require several subsequent steps; real world problems may require a substantial amount of exploration. To investigate this issue, we will employ a problem that *requires* exploration.

### 6.3 The Discretized COMPARE-ON-ONE Problem

Since we expect exploration to be a necessary ingredient for real-world coevolutionary problems, a question is how reliability can be combined with exploration. In order to test the ability of coevolution algorithms to perform exploration, we describe a new test problem called the *Discretized* COMPARE-ON-ONE *Problem*.

In the discretized COMPARE-ON-ONE problem (see Figure 3), the value in each dimension of an individual is rounded to the nearest multiple of a parameter $\delta$ below it. This discretization is applied to both candidates and tests before evaluating the outcome of the problem, without affecting the genotype.

$$m = \arg\max_i d(T_i) \tag{4}$$

$$\text{discretized compare} - \text{on} - \text{one}: \quad G(C,T) = \begin{cases} 1 & \text{if} \quad d(C_m) \geq d(T_m) \\ -1 & \text{otherwise} \end{cases} \tag{5}$$

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

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

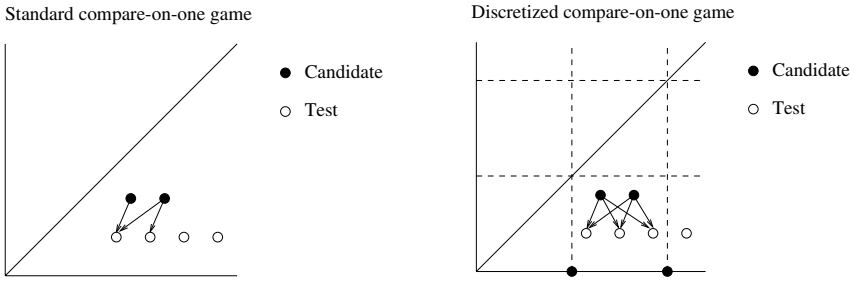


Figure 3: Left: the standard COMPARE-ON-ONE game. Candidates are tested on the highest dimension of the test. Thus, tests below the diagonal test the horizontal dimension of candidates. Comparison of the tests solved by candidates (see arrows) provides a gradient indicating the direction of improvement. Right: The discretized COMPARE-ON-ONE game. All individuals are mapped to the lower-left corner of their square in the discretization grid. As a result, the candidates solve all tests in their square, and much of the gradient information is lost.

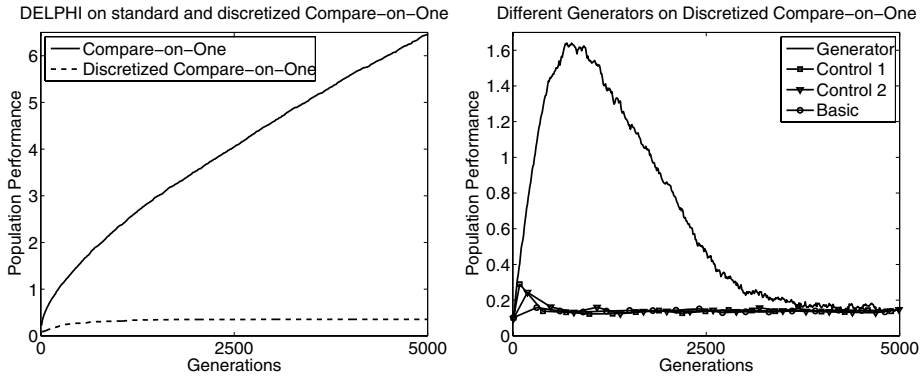


Figure 4: Left: DELPHI on the standard and discretized COMPARE-ON-ONE problem. All results are averaged over 50 runs. Right: Comparison between different generators. None of the methods can reliably address the discretized COMPARE-ON-ONE problem when used stand-alone.

Since a test in DELPHI can only be replaced by its own offspring, exploration is very limited. DELPHI would therefore be expected to fail on the discretized COMPARE-ON-ONE problem. We compare the performance of the algorithm on the standard and discretized version of the problem, see Fig. 4 (left). The results confirm the expectation that the discretized COMPARE-ON-ONE problem cannot be addressed by an algorithm with little capacity for exploration, thus making the problem suitable for testing the explorative capacity of coevolutionary algorithms.

### 6.4 DELPHI versus IPCA

The IPCA algorithm addresses two shortcomings of DELPHI. First, as has been shown, any change to the archive constitutes an improvement, and reliability is thereby guaranteed. Moreover, since IPCA is an archive method, any individual can be presented to the archive, regardless of how it was produced or of its relation to the current members of the archive. Therefore, IPCA can be combined with a highly explorative *generator* of new individuals. For example a highly stochastic coevolutionary algorithm may be used, since the generator does not need to be reliable itself.

An interesting possibility is to use the reliability of the archive to make the generator more reliable. This can be done by periodically copying archive members back to the generator population, or by using archive members as parents when generating new individuals for the generator population. The archive then functions as a *ratchet* that maintains the performance level of the generator and prevents it from regressing, as will be seen in experiments.

### 6.5 Generators

We now describe the generators that will be used in the experiments. The generators are explorative coevolutionary methods that have some potential for maintaining diverse tests. Since the reliability of these basic coevolution algorithms is not guaranteed, an archive method is needed to ensure overall progress. The generators are tested both as stand-alone methods and in combination with archive methods.

All generators that will be used consist of a candidate population and a test population, both of size 50. New individuals for each population are created in a generational setup. If the generator is used in combination with an archive method, parents are selected randomly from the archive with $P = 0.05$. Otherwise, parents are selected randomly from the respective population. New individuals are created using two-point crossover (50%, where used) and mutation (50%).

To mutate the value in a dimension, a value randomly chosen from $[-\frac{r}{2} - b, \frac{r}{2} - b]$ is added, where $r$ is the mutation range and $b$ the mutation bias. In the experiments, $r = .25$ and $b = .025$, yielding a range of $[-.15, .1]$ for the added random constant.

Mutation is always applied to two randomly chosen dimensions. This makes the problem more difficult than if a single dimension would be mutated (De Jong & Pollack, 2004), since an improved outcome against an opponent no longer guarantees that the individual has improved overall. In fact, given that at least one of the dimensions has increased, the other mutated dimension is substantially more likely to have decreased than increased.[2]

Once new individuals have been generated, the current populations and new generations of candidates and tests are both merged and evaluated, and 50% of the individuals are selected to make up the new populations.

[2]Specifically, given that at least one of the dimensions has increased (p=0.64), the probability that the other has decreased is $\frac{0.48}{0.64} = 0.75$.

The objectives for candidates are their outcomes against the tests in the test-population, and vice-versa. In addition, tests can use the *distinctions* between candidates as objectives. A test makes a distinction (Ficici & Pollack, 2001b) between candidates $C_i$ and $C_j$ if it assigns a higher score to $C_i$ than to $C_j$:

$$dist(T, C_1, C_2) \iff G(C_1, T) > G(C_2, T).$$

Since the number of objectives is large, we do not employ a dominance-based selection method, but calculate a scalar fitness value based on the objective values. Specifically, all outcomes against individuals in the other population are added, yielding a sum of scores. If distinction objectives are used, the number of distinctions made is calculated too. A weighted average of these two scores is then taken, where the sum of the scores has a relative weight of $w_{score} = 0.75$ versus $1 - w_{score} = 0.25$ for the number of distinctions.

In some of the methods, a fitness sharing technique similar to Rosin's *competitive fitness sharing* method (1997) is used. Each point (a positive outcome against an opponent, or a distinction) is divided by the total number of individuals that receive the point. The final fitness values are used to perform fitness-proportional selection without replacement. If the sum of the fitness values falls below a near-zero threshold, the remaining individuals are selected randomly. The selected individuals replace the current population.

To facilitate reproduction of our results, we summarize the operation of the generator in combination with the IPCA archive; see pseudo-code in Fig. 5. The candidate and test populations of the generator are initialized to contain $popsize = 50$ randomly generated individuals each. Next, the following loop is repeated until a stop criterion is reached. First, $popsize$ new candidates and tests are generated. These are submitted to the archive (IPCA or LAPCA), upon which the archive updates its candidate and test archives according to the algorithm that has been described. Finally, the generator evaluates its candidate and test populations, and performs selection.

Evaluate_and_replace operates as follows. First, the current populations (Cset and Tset) are merged with the newly generated individuals (Cnew and Tnew). Next, scores are computed for the combined candidate and test sets. Finally, fitness proportional selection without replacement is applied to each set, based on the computed scores.

The score for a candidate solution is computed as follows: for each test, a point is added if the outcome of the candidate solution against the test is positive. If fitness sharing is used, the point is divided by the number of candidate solutions that have the point.

For a test, the assignment of scores is analogous. Additionally though, if distinctions are used, points for distinctions between candidate solutions are added; a point is added for each pair of candidate solutions between which the test makes a distinction, again optionally divided by the number of tests that make the distinction if fitness sharing is used. This concludes the description of the setup.

The generator used in combination with IPCA uses diversity objectives and fitness sharing, and will simply be called generator. As a comparison, a basic coevolutionary method (basic) is used; this method is equal to generator except that it does not use distinction objectives or fitness sharing. We also employ control methods control1 and control2, each representing one of the two differences between basic and generator. Table 1 summarizes the properties of these methods (left) and their parameters (right). For all experiments, results are averaged over 50 runs.

```
MAINLOOP(){
  generator.fill − populations();
  t := 0
  archive.CS^t := ∅
  archive.TS^t := ∅
  while ¬ stop − crit
    Cnew := generator.generate_candidates(archive.CS^t);
    Tnew := generator.generate_tests(archive.TS^t);
    t := archive.submit(Cnew, Tnew, t);
    generator.evaluate_and_replace(Cnew, Tnew);
  end
}
```

Figure 5: Summary of the overall setup used in the experiments, comprising the generator (a two-population coevolutionary algorithm) and the archive (IPCA or LAPCA).

| | Distinction objectives | Fitness sharing |
|---|---|---|
| basic | - | - |
| control1 | + | - |
| control2 | - | + |
| generator | + | + |

| Parameter | Value |
|---|---|
| popsize | 50 |
| $r$ | .25 |
| $b$ | .025 |
| $w_{score}$ | .75 |
| $\delta$ | .25 |

Table 1: Left: Properties of the four generators used in the experiments. Right: Summary of the parameters used in the experiments.

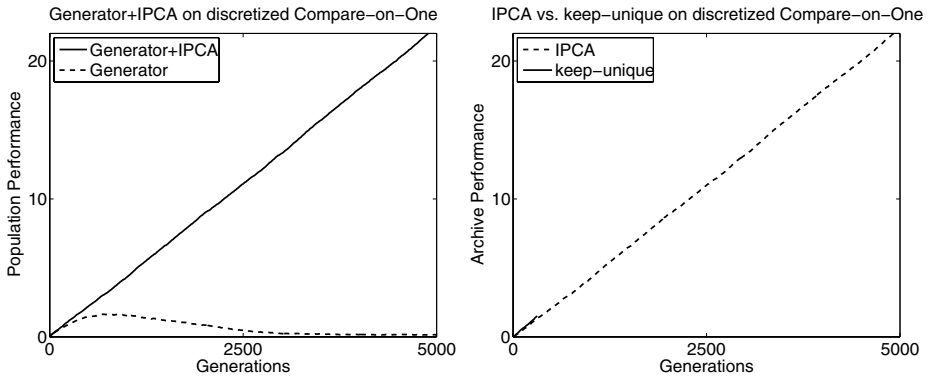


Figure 6: Left: When Generator is used in combination with IPCA, the archive functions as a ratchet, and stable progress is achieved. Right: Performance of IPCA vs. keep-unique. keep-unique obtains a marginally higher performance increase per generation, but can only be run for a small number of generations due to its quickly growing archive.

### 6.6 Results with IPCA

We will now evaluate IPCA in experiments. To avoid unnecessary evaluations, all game-outcomes are cached, and thus calculated and counted only once.

First, we compare the four different generators. Figure 4 (right) shows the results. It is seen that generator performs better than the three other methods. However, when used as a stand-alone method, generator is unable to address the discretized COMPARE-ON-ONE problem due to its lack of reliability; after an initial period where performance increases steadily, a sharp drop occurs, indicating the algorithm has lost the ability to test on all three underlying objectives. As a result, the performance of candidates in the untested dimension(s) is driven down by the negative mutation bias.

To investigate whether the results of the stochastic generator can be improved by using a reliable archive, we compare the performance of generator with its performance when combined with IPCA. Figure 6 (left) shows the performance of the generator population. When used stand-alone (same curve as in Figure 4, right, but different scale), the generator is unable to address the discretized COMPARE-ON-ONE problem. By using the generator in combination with IPCA however, stable progress is made on the problem. The only difference between these two setups regarding the generator is that in the latter case, archive members are occasionally used as parents for generating new individuals. Thus, we can conclude that the archive not only serves to maintain valuable candidates and informative tests encountered so far, but it can also function as a catalyst that improves the performance of a given coevolutionary algorithm.

To verify that the difference in performance is due to the use of archive parents, we also used the setup with the generator and IPCA without archive parents. In this case, the archive merely accepts individuals from the generator, and cannot influence its behavior. The performance of the generator population in this case was identical to that for the stand-alone generator, as expected.

#### 6.6.1 Receptivity of IPCA

To evaluate the receptivity of IPCA, we measure its performance in terms of the number of *generations* of the generator, and compare this against a maximally receptive archive. The simplest maximally receptive archive is a method that accepts *all* individuals, except identical ones. Since dominated candidates can be safely discarded, we employ a method called keep-unique that maintains all non-dominated candidates and accepts all unique tests, i.e. all new tests for which no existing test in the archive has identical genotypic values. Figure 6 (right) shows that keep-unique obtains only marginally better performance when viewed in terms of the number of generations. This indicates that IPCA's receptivity is close to optimal.

#### 6.6.2 Selectivity of IPCA

To evaluate the selectivity of an archive, the question of interest is how many individuals may be safely removed without jeopardizing progress towards the solution concept. Selectivity is minimal when all individuals are maintained. We therefore use the same comparison method, (keep-unique), and consider the development of the test archive sizes. As Figure 7 (left) shows, IPCA greatly reduces the number of tests that are maintained compared to the baseline of keep-unique.

#### 6.6.3 Efficiency of IPCA

The ultimate evaluation criterion for an archive is its efficiency, i.e., the rate of progress it establishes, measured as a function of computation time. To evaluate the efficiency of

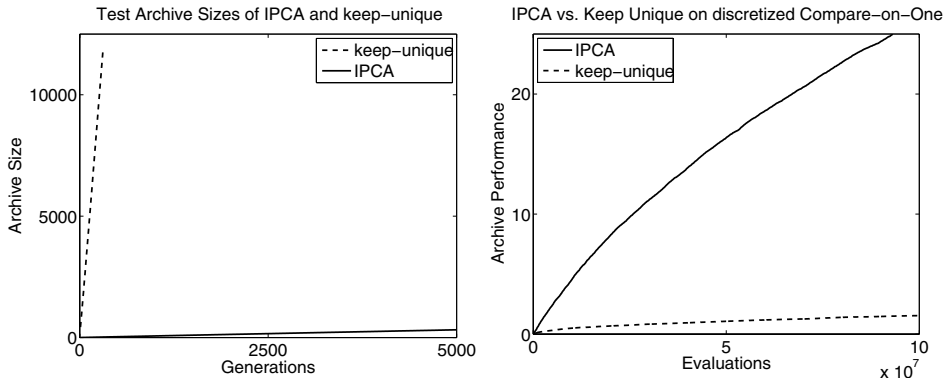


Figure 7: Left: Test archive sizes for IPCA and keep-unique. Right: Plotting the results in terms of the number of *evaluations* reveals that IPCA progresses much quicker than keep-unique, due to its efficient archiving strategy.

IPCA, we measure performance as a function of the number of evaluations performed. Figure 7 (right) shows the results.

When viewed in terms of evaluations, it is revealed that IPCA achieves much better performance as a function of computational effort than keep-unique. This indicates that the IPCA acceptance criterion is effective at separating relevant individuals from those that may be safely discarded.

In summary, IPCA has close to optimal receptivity, and its selectivity is greatly improved compared to the baseline of keeping all unique tests. A remaining question is whether a more efficient archive can be designed by further increasing selectivity. We consider this question below.

## 7 Towards a Bounded Pareto-Coevolution Archive

The IPCA algorithm avoids the problems of cycling and regress that can occur in coevolutionary setups. It is first and foremost of theoretical interest, as the growing test archive can limit its scope of application. We therefore investigate whether a more scalable algorithm can be devised that still provides reasonable reliability.

### 7.1 The LAyered Pareto-Coevolution Archive: LAPCA

In Evolutionary Multi-Objective Optimization, a selection technique exists that maintains a number of *layers* in addition to the Pareto front; this technique is used for example in the NSGA algorithm (Srinivas & Deb, 1994). The LAyered Pareto-Coevolution Archive (LAPCA) is based on this idea, and maintains the first $n$ layers of candidates. These layers are determined as follows. The first layer is the Pareto front itself, and consists of the nondominated candidates. The second layer is obtained by removing the first layer from the population and determining which of the remaining individuals have become nondominated by removing the previous layer. Subsequent layers are obtained by repeating this same procedure. In order to avoid spurious growth of the archive, if there are multiple individuals in a layer with identical outcome vectors (i.e. identical outcomes against all tests), only one of these individuals is maintained.

The test archive in LAPCA is updated as follows. For any two subsequent layers of candidates, a *separation set* is determined and selected to form part of the updated

version of the archive. The updated test archive *only* contains the union of these separation sets; any previous tests that are not used in any separation set will be discarded. The separation set is defined as follows. For any candidates $C_i, C_j$ in layers $i$ and $j$ where $i = j$ or $i = j - 1$, if any available test makes a distinction (see earlier) between $C_i$ and $C_j$, then the separation set must contain a test that makes this distinction.

In determining which of the existing and new tests are to be maintained, tests are visited in the given order in which they arrive, and the first test that makes a distinction is marked to be selected. Thus, if some tests make multiple distinctions, this can reduce the number of required tests.

### 7.2 Reliability of Layered Approaches

An interesting question is how the number of layers in a layered approach affects the reliability of the archive. First we observe that two layers can already provide a substantially higher degree of reliability than a single layer. In a single layer, the arrival of a single candidate that dominates all existing candidates removes all those other candidates from the archive. As a result, no distinctions can be made by the tests, and thus the test archive is emptied. Notwithstanding its high quality, the candidate is now prone to removal from the archive; as soon as a candidate arrives that has a higher score for even one test, it could be removed as it would appear dominated in the presence of only this test.

When a second layer is added, this layer acts as a buffer separating nondominated from dominated individuals; by maintaining candidates of lower quality, tests will be maintained that distinguish between the candidates in the first and second layer, and newly arrived candidates will be evaluated on the aspects represented by these tests.

By further increasing the number of layers, reliability can be further increased. In earlier work, we have identified counter-examples in which reliability was not guaranteed for one, two, and three layers (De Jong, 2004b). It was found that such examples quickly become more involved and rare with an increasing number of layers, implying that the reliability of the archive grows rapidly with the number of layers.

A more general question is whether any number of layers is sufficient to guarantee reliability. We will show that this is not the case. For any finite number of layers $n$, a counter-example against the reliability of the archive can be constructed as follows (see Figure 8).

Let $C_1, \ldots, C_{n+1}$ be candidates and $T_1, \ldots, T_n$ be tests. Let the outcomes of the candidates on these tests be chosen such that $C_i$ solves tests $T_1, \ldots, T_{i-1}$. Thus, $C_2$ solves $T_1$; $C_3$ solves $T_1$ and $T_2$; etc. Since each candidate dominates its predecessor, each candidate forms its own layer. Thus, there are $n + 1$ layers. Only $n$ layers are maintained, so candidate $C_1$ will be discarded. Test $T_1$ is now solved by all remaining candidates, meaning it does not make any distinctions, and will be removed accordingly. Now that the archive has lost the ability to test candidates on whether they solve $T_1$, newly arrived candidates that do not solve $T_1$ can dominate the existing candidates and cause them to be removed. Thus, the ability to solve $T_1$ can be lost. This demonstrates that unlike IPCA, LAPCA cannot truly guarantee monotonicity, even if it may closely approximate it. Since LAPCA's reliability grows quickly with the number of layers, it may be sufficiently reliable in practice, and the efficiency of a limited archive size may weigh up against the lack of monotonicity guarantee. To determine how LAPCA trades the reliability guarantee for efficiency, we now investigate the method in experiments.

| | T1 | T2 | T3 | .. | $T_n$ |
|---|---|---|---|---|---|
| L1 | 0 | 0 | 0 | . | 0 |
| L2 | 1 | 0 | 0 | . | 0 |
| L3 | 1 | 1 | 0 | | 0 |
| L4 | 1 | 1 | 1 | | 0 |
| .. | . | . | | | |
| $L_{n+1}$ | 1 | 1 | 1 | 1 | 1 |

| | T1 | T2 | T3 | .. | $T_n$ |
|---|---|---|---|---|---|
| L1 | 0 | 0 | 0 | . | 0 |
| L2 | 1 | 0 | 0 | . | 0 |
| L3 | 1 | 1 | 0 | | 0 |
| L4 | 1 | 1 | 1 | | 0 |
| .. | | . | | | |
| $L_{n+1}$ | 1 | 1 | 1 | 1 | 1 |

Figure 8: Counter-example, demonstrating that no finite number of layers can guarantee reliability. $C_1$ forms layer $n+1$, and is therefore removed. Since $T_1$ then no longer makes any distinctions, it is removed too, so that the archive loses its ability to perform this test. As a result, all candidates solving $T1$ may eventually be replaced, so that the archive loses the ability to solve $T1$.

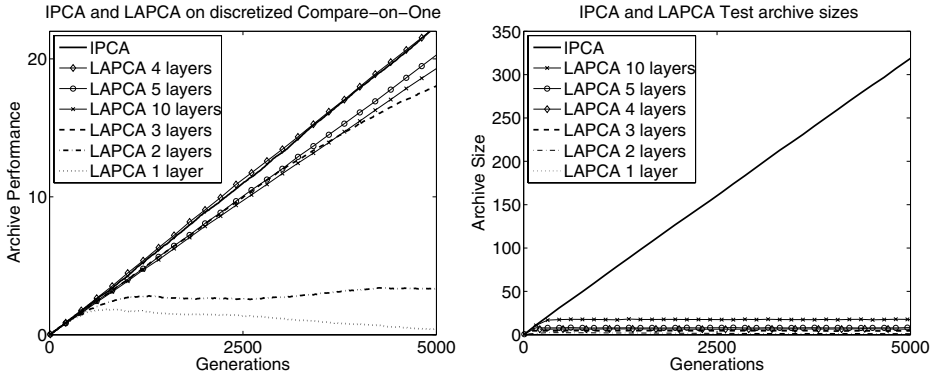


Figure 9: Left: Performance of IPCA and LAPCA on the discretized COMPARE-ON-ONE problem as a function of generations. For a sufficient number of layers, LAPCA also achieves reliable progress. Right: Test archive sizes for the same methods. While the test archive for IPCA grows approximately linearly, LAPCA features stable and limited archive sizes.

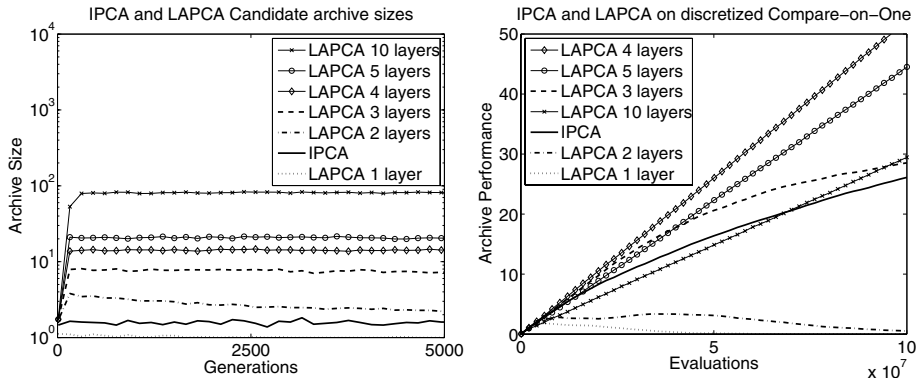


Figure 10: Left: Average size of the candidate archives for IPCA and LAPCA. For every number of layers tried, the archive sizes stabilize over time. Right: The same results as in Figure 9 (left), now plotted as function of evaluations. Using a sufficient number of layers, LAPCA is more efficient than IPCA due to its limited archive sizes, while also effectively providing reliability.

### 7.3 Results with LAPCA

#### 7.3.1 Receptivity of LAPCA

Figure 9 (left) shows the performance of the LAPCA candidate archive as a function of the number of generations. One or two layers are insufficient to establish reliable progress. For three layers, performance progresses almost linearly, but a slight decline sets in towards the end of the experiment, around 4,000 generations. For four, five, or ten layers, LAPCA is sufficiently reliable, and its performance is comparable to that of IPCA. This implies LAPCA's receptivity for these latter parameters is comparable to that of IPCA.

The increased reliability explains the initial increase in performance for an increasing number of layers. Once the number of layers is sufficient to produce reliable progress however, adding more layers decreases the performance, even when viewed in terms of the number of generations. Our explanation for this is that the quality of archive members selected to function as parents degrades with an increasing number of layers, as larger archives contain more low-quality candidates.

#### 7.3.2 Selectivity of LAPCA

An important difference between IPCA and LAPCA is that the latter has limited archives. Thus, it is expected to be much more selective. Figure 9 (right) compares the test archive sizes of IPCA and LAPCA. The graph reveals the clear qualitative difference between IPCA and LAPCA: whereas the size of IPCA's test archive grows linearly over time, the archive sizes for LAPCA stabilize and remain small.

The sizes for the candidate archives are shown in figure 10 (left). All methods quickly reach a reasonably stable archive size, which is determined by the sizes of the Pareto-front and subsequent layers. For LAPCA there is a clear correspondence between the number of layers and the archive size.

### 7.3.3 Efficiency of LAPCA

LAPCA has been seen to be comparable in receptivity to IPCA for an intermediate or high number of layers. Since LAPCA also has a higher selectivity particularly for an intermediate number of layers, the method is expected to be more efficient overall. To verify this, we plot the performance of both archive methods in terms of the number of evaluations.

Figure 10 (right) clearly shows the trade-off between reliability and efficiency offered by LAPCA. The curves for LAPCA with three and ten layers end slightly above IPCA, while with four and five layers LAPCA is substantially more efficient than IPCA. To verify whether these differences are statistically significant, we have performed the non-parametric Wilcoxon rank sum test based on the archive performance at the end of each run, i.e. after $10^8$ evaluations. The difference between IPCA and LAPCA with three layers is not significant. The differences between IPCA and LAPCA with four, five, and ten layers however are all significant at a confidence level of $\alpha = .001$.

The graph furthermore shows that for the most reliable variants (five and ten layers), performance increases linearly in the number of *evaluations*, suggesting that LAPCA provides a scalable approach to the difficult discretized Compare-on-One problem.

### 7.4 Discussion

A main open issue with regard to practical applications is that the size of the Pareto-optimal set can be very large; this is a general problem in Evolutionary Multi-Objective Optimization, and general solutions apply. A possible approach to this is to select a limited number of directions along which progress is to be made; the increased number of possibilities for improvement may aid in circumventing local optima compared to any single weighting of the different objectives such as the average test outcome.

## 8 Conclusions

In this article we have presented an algorithm, named IPCA, that guarantees monotonic progress in coevolution based on the solution concept of the Pareto-optimal Equivalence Set. IPCA was found to make consistent progress on the challenging discretized COMPARE-ON-ONE problem, which could not be addressed by standard coevolution methods used for comparison. Due to its unlimited archive size, IPCA is mainly of theoretical interest. Therefore, a layered variant named LAPCA has also been described. This algorithm features a tunable trade-off between reliability and efficiency, and performs more efficiently in practice.

The contributions of this article are as follows. First, the notion of monotonicity for solution concepts in coevolution has been defined in a precise sense. Next, the monotonic IPCA algorithm was presented. It is proven that IPCA guarantees monotonic progress with respect to the solution concept of the Pareto-optimal Equivalence Set. To achieve a limited archive size, the layered LAPCA algorithm has been described. It was shown that no number of layers is sufficient to guarantee monotonicity; yet, in experiments, the performance of this algorithm was found to be higher than that of IPCA, and the algorithm is therefore of interest from a practical perspective.

Three features have been introduced that can be used to evaluate coevolutionary archive methods: receptivity, selectivity, and efficiency. The features have been applied to evaluate the IPCA and LAPCA methods. It was found that IPCA achieves near-optimal receptivity, and a substantial degree of selectivity. LAPCA was found to provide a qualitatively higher selectivity than IPCA, and a comparable receptivity

when a sufficient number of layers is used. As a result, LAPCA's overall efficiency is higher than that of IPCA for an intermediate number of layers.

From a practical standpoint, the consequences of these contributions are as follows. First, the LAPCA algorithm can be applied in practice to coevolutionary problems. For problems where the lack of monotonicity of an existing coevolutionary algorithm hampers performance, any existing coevolution algorithm may be combined with LAPCA to yield a setup that approximates monotonicity. Second, the IPCA and LAPCA algorithms may serve as a starting point to develop more efficient monotonic or near-monotonic coevolutionary algorithms. Third, at a meta level, the work that has been described exemplifies an approach that may be usefully applied in any area of evolutionary computation, and can be summarized as follows. At the outset, the properties that characterize some ideal desired behavior are formally specified. Next, an algorithm is devised that guarantees the desired behavior, but may suffer from practical limitations. Finally, a variant of the theoretical algorithm is designed that approximates the desired guarantees, but does so with limited resources, so that an efficient algorithm is obtained that provides the desired behavior in practice.

An open issue is how an archive method can best be combined with an existing coevolutionary method that serves as a generator. As discussed, the archive can not only be used to maintain valuable individuals, but due to its monotonicity, it can also serve to put the population back on track when performance has dropped. One approach to do this is to use archive members as parents. However, other methods may be more effective or efficient; research into this question would be valuable.

A further open issue in coevolution research is to provide bounds on the convergence time or time to solution for coevolutionary archive methods. Also, the transfer of the techniques for archive size control used in Evolutionary Multi-Objective Optimization may be investigated. Finally, since each solution concept has its own particular advantages and disadvantages, the development of monotonic archives for other solution concepts would be valuable.

## Symbols

| | |
|---|---|
| $\mathbb{C}$ | the set of all possible candidate solutions |
| $\mathbb{T}$ | the set of all possible tests |
| $T$ | a test |
| $C$ | a candidate |
| $G(C,T)$ | the outcome of $C$ against $T$ according to interaction function $G$ |
| $S = (CS, TS)$ | an approximation of a solution concept |
| $CS$ | a set of candidates |
| $TS$ | a set of tests |
| $S_C$ | the candidate set of $S$ |
| $S_T$ | the test set of $S$ |
| $S^t$ | the approximation of the solution concept at time $t$ |
| $S^1, S^2, \ldots S^n$ | a sequence of approximations to the solution concept |
| $CS^1, CS^2, \ldots CS^n$ | the sequence of candidate archives obtained over time |
| $TS^1, TS^2, \ldots TS^n$ | the sequence of test archives obtained over time |
| $\succ$ | the Pareto-dominance relation |
| $D$ | distance function |
| $r$ | mutation range |
| $b$ | mutation bias |

## Acknowledgments

The author wishes to thank the members of the DSS and LDD Groups at Utrecht University for a pleasant working environment, and Paul Wiegand, Anthony Bucci, and the anonymous reviewers for valuable feedback on this work.

## Appendix A

**Proof 1 (Monotonicity)** *We show that a sequence for which every transition satisfies the transition criterion of Def. 2 is monotonic according to Def. 1. Assume that each transition between two subsequent elements in a sequence of approximations* $\{S^1, S^2, \ldots S^n\}$ *satisfies the transition criterion. Thus, for every* $t, 1 \leq t \leq n-1$*:*

$$\begin{aligned} \forall TS \subseteq S_T^t: \quad & [\exists C \in S_C^t : solves(C, TS) \implies \\ & \exists C' \in S_C^{t+1} : solves(C', TS)] \qquad (6) \\ \exists TS \subseteq S_T^{t+1}: \quad & [\nexists C \in S_C^t : solves(C, TS) \wedge \\ & \exists C' \in S_C^{t+1} : solves(C', TS)]. \qquad (7) \end{aligned}$$

*It is to be shown that this implies that for every* $t, 1 \leq t \leq n-1 : D(S^{t+1}, (Z, \mathbb{T})) < D(S^t, (Z, \mathbb{T}))$*. Eq. 6 states that any subset of* $S_T^t$ *solved by* $S_C^t$ *is also solved by* $S_C^{t+1}$*. Thus, the* $TS()$ *measure at least remains equal during a transition from* $t$ *to* $t+1$*:*

$$|TS(S^{t+1})| \geq |TS(S^t)|.$$

*Eq. 7 states that in addition, there exists a candidate* $C' \in S_C^{t+1}$ *that solves a test set* $TS$ *not solved by any candidate* $C \in S_C^t$*. This guarantees that the* $TS()$ *measure must increase:*

$$|TS(S^{t+1})| > |TS(S^t)|. \qquad (8)$$

*To determine how this affects the distance to the solution concept* $Z$*, we first observe that the solution concept must solve at least the same number of test sets as any approximation of the solution concept:*

$$|TS(Z, \mathbb{T})| \geq |TS(S^{t'})|,$$

*where* $t'$ *can be e.g.* $t$ *or* $t+1$*. This fixes the sign of the distance function given in Eq. 1, so that the distances to the solution concept at times* $t$ *and* $t+1$ *are given by:*

$$\begin{aligned} D(S^{t+1}, (Z, \mathbb{T})) &= |TS(Z, \mathbb{T}))| - |TS(S^{t+1})| \\ D(S^t, (Z, \mathbb{T})) &= |TS(Z, \mathbb{T}))| - |TS(S^t)|. \end{aligned}$$

*Comparing these distances and combining with Eq. 8 yields:*

$$D(S^{t+1}, (Z, \mathbb{T}))) < D(S^t, (Z, \mathbb{T})).$$

*Thus, for any transition from* $t$ *to* $t+1$*, the distance to the solution concept decreases, and the sequence is monotonic according to Def. 1.* ■

## Appendix B

**Proof 2 (Convergence)** *We prove that if the IPCA algorithm has not reached the solution concept $S4$, a step that brings the algorithm closer to the solution concept can always be made.*

*Assume that the solution concept $S4$ has not yet been reached. Then according to the definition of $S4$, there must exist a set of tests $TS \subseteq \mathbb{T}$ that can be solved by some candidate but is not solved by any candidate in the current candidate archive $CS^t$:*

$$\exists C \in \mathbb{C} : solves(C, TS) \quad \wedge \quad \nexists C' \in CS^t : solves(C', TS).$$

*Since no candidate in the archive solves $TS$, $C$ is guaranteed to be non-dominated and unique, and thus satisfies the* useful *criterion. The candidate would therefore be accepted by IPCA. Thus, supplying $C$ and $TS$ to the archive guarantees that a step can be made.* ■

## References


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

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

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

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

Bucci, A., & Pollack, J. B. (2002). Order-theoretic analysis of coevolution problems: Coevolutionary statics. In Barry, A. M. (Ed.), *Proceedings of the GECCO-02 Workshop on Coevolution: Understanding Coevolution*, pp. 229–235. AAAI.

Bucci, A., & Pollack, J. B. (2003a). Focusing versus intransitivity. Geometrical aspects of coevolution.. In Cantú-Paz, E., et al. (Ed.), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-03*, Berlin. Springer.

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

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 Kalyanmoy Deb et al. (Ed.), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-04*, pp. 501–512. Springer.

Cartlidge, J., & Bullock, S. (2004). Combating coevolutionary disengagement by reducing parasite virulence.. *Evolutionary Computation*, *12*(2), 193–222.

De Jong, E. D. (2003). Representation development from Pareto-coevolution. In Cantú-Paz, E., et al. (Ed.), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-03*, pp. 262–273, Berlin. Springer.

De Jong, E. D. (2004a). The Incremental Pareto-Coevolution Archive. In Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E. K., Darwen, P. J., Dasgupta, D., Floreano, D., Foster, J. A., Harman, M., Holland, O., Lanzi, P. L., Spector, L., Tettamanzi, A., Thierens, D., & Tyrrell, A. M. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-04*, pp. 525–536. Springer.

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

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.

De Jong, K. A. (2006). *Evolutionary Computation. A unified approach.* A Bradford Book. The MIT Press.

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., Melnik, O., & Pollack, J. B. (2005). A game-theoretic and dynamical-systems analysis of selection methods in coevolution. *IEEE Transactions on Evolutionary Computation*, *9*(6), 580–602.

Ficici, S. G., & Pollack, J. B. (1998). Coevolving communicative behavior in a linear pursuer-evader game. In Pfeifer, R., Blumberg, B., Meyer, J.-A., & Wilson, S. (Eds.), *From Animals to Animats. Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior, SAB-98*, pp. 263–269, Cambridge, MA. The MIT Press.

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. (2001a). Game theory and the simple coevolutionary algorithm: Some preliminary results on fitness sharing. In Belew, R. K., & Juillé, H. (Eds.), *Proceedings of the GECCO-01 Workshop on Coevolution: Turning Adaptive Algorithms upon Themselves*, pp. 2–7, San Francisco, CA. Morgan Kaufmann.

Ficici, S. G., & Pollack, J. B. (2001b). Pareto optimality in coevolutionary learning. In Kelemen, J. (Ed.), *Sixth European Conference on Artificial Life*, pp. 316–325, 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.

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, New York.

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

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Boston, MA.

Goldberg, D. E. (2002). *The design of innovation. Lessons from and for competent genetic algorithms.* Kluwer Academic Publishers, Norwell, MA, USA.

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

Holland, J. H. (1975). *Adaptation in Natural and Artifical Systems*. University of Michigan Press, Ann Arbor, MI.

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

Jansen, T., & Wiegand, R. P. (2004). The cooperative coevolutionary (1+1) EA. *Evolutionary Computation*, *12*(4), 405–434.

Juillé, H. (1999). *Methods for Statistical Inference: Extending the Evolutionary Computation Paradigm*. Ph.D. thesis, Brandeis University.

Juillé, H., & Pollack, J. B. (1996). Co-evolving intertwined spirals. In Fogel, L., Angeline, P., & Baeck, T. (Eds.), *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pp. 461–467, Cambridge, MA. The MIT Press.

Juillé, H., & Pollack, J. B. (1998). Coevolving the "ideal" trainer: Application to the discovery of cellular automata rules. In Koza, J. R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M. H., Goldberg, D. E., Iba, H., & Riolo, R. (Eds.), *Proceedings of the Third Annual Genetic Programming Conference*, pp. 519–527, San Francisco, CA, USA. Morgan Kaufmann.

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

Knowles, J., & Corne, D. (2004). Bounded Pareto Archiving: Theory and Practice. In Gandibleux, X., Sevaux, M., Sörensen, K., & T'kindt, V. (Eds.), *Metaheuristics for Multiobjective Optimisation*, pp. 39–64, Berlin. Springer. Lecture Notes in Economics and Mathematical Systems Vol. 535.

Knowles, J., Corne, D., & Fleischer, M. (2003). Bounded archiving using the lebesgue measure. In Sarker, R., Reynolds, R., Abbass, H., Tan, K. C., McKay, B., Essam, D., & Gedeon, T. (Eds.), *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pp. 2490–2497, Canberra. IEEE Press.

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

Laumanns, M., Thiele, L., Deb, K., & Zitzler, E. (2002). Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, *10*(3), 263–282.

Liekens, A. M. L., ten Eikelder, H. M. M., & Hilbers, P. A. J. (2003). Finite population models of co-evolution and their application to haploidy versus diploidy. In Cantú-Paz, E., Foster, J. A., Deb, K., Davis, D., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R., Kendall, G., Wilson, S., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K., Jonoska, N., & Miller, J. (Eds.), *Genetic and Evolutionary Computation – GECCO-2003*, Vol. 2723 of *LNCS*, pp. 344–355, Chicago. Springer-Verlag.

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

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

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

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

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

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. Complex Adaptive Systems. Cambridge, MA, The MIT Press.

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

Noble, J., & Watson, R. A. (2001). Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for Pareto selection. In Spector, L., et al. (Ed.), *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 493–500, San Francisco, CA. Morgan Kaufmann.

Osborne, M. J., & Rubinstein, A. (1994). *A course in game theory*. Cambridge, MA, The MIT Press.

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

Paredis, J. (1996). Coevolutionary computation. *Artificial Life*, *2*(4), 355–375.

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

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

Rechenberg, I. (1994). *Evolutionsstrategie '94*. frommann-holzbog, Stuttgart. German.

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

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

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

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

Sims, K. (1994). Evolving 3D morphology and behavior by competition. In Brooks, R., & Maes, P. (Eds.), *Artificial Life IV*, pp. 28–39, Cambridge, MA. The MIT Press.

Srinivas, N., & Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, *2*(3), 221–248.

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

Watson, R. A., & Pollack, J. B. (2000). Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In Schoenauer, M., et al. (Ed.), *Parallel Problem Solving from Nature, PPSN-VI*, Vol. 1917 of *LNCS*, pp. 425–434, 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.

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

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

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

Wiegand, R. P., Liles, W. C., & De Jong, K. (2003). Modeling variation in cooperative coevolution using evolutionary game theory. In De Jong, K. A., Poli, R., & Rowe, J. E. (Eds.), *Foundations of Genetic Algorithms 7*, pp. 203–220. Morgan Kaufmann, San Francisco.