Order-theoretic Analysis of Coevolution Problems: Coevolutionary Statics

Anthony Bucci and Jordan B. Pollack
DEMO Laboratory, Computer Science Department, MS018
Brandeis University
Waltham, MA 02454
{abucci,pollack}@cs.brandeis.edu

Abstract

We present an order-theoretic framework for analyzing coevolution problems. The framework focuses attention on the underlying problem definition, or *statics* of coevolution, as opposed to the dynamics of search algorithms. We define a notion of solution for coevolution which generalizes similar solution concepts in GA function optimization and MOO. We then define the ideal test set, a potentially small set of tests which allow us to find the solution set of a problem. One feature of the ideal test set is that we are able to categorize problems by considering its cardinality. We conclude by discussing three issues which commonly arise in coevolution from the point of view of coevolutionary statics, pointing out analytical attacks on these issues.

1 Introduction

Over the past decade, the field of coevolutionary optimization has developed tantalizing learning and optimization results in domains such as sorting networks [7], cellular automata [5, 8], game playing [11] and robotics [13]. However, to date the field has progressed using problem-specific tricks and heuristics, leaving the field with a relative lack of theory. One reason for the current state of affairs is that the dynamics of coevolution can be complex and difficult to understand. Because this difficulty, we propose a change in point of view, approaching the study of coevolution in terms of the static structure of the underlying problem definition. We argue that by studying the problem definition, prior to any algorithm choices, we might gain insight which is not clear from studying algorithm dynamics alone.

Our tool for this study will be a mathematical framework based on order relations over the sets of candidates and tests, which we will detail in section 3. Our framework focuses attention on the definition of the coevolution problem and reveals theoretical order relations which exist among the candidate solutions and among the tests we are able to apply to them.

We will consider a class of coevolutionary optimization problems which can be expressed with a function of the form $p : S \times T \to R$ where S and T are sets and R is an ordered set. We intend S to represent the candidate solutions to the problem; these are also called students or learners in the coevolutionary learning context. T represents tests which can be applied to the candidates. The ordered set R contains the outcomes of a candidate-test interaction. The ordering tells us which outcomes are better than which others. For example,

Example 1.1 *Rock-paper-scissors*

In this simple game, $S = \{rock, paper, scissors\}$, $T = S$ and $R = \{0 < 1\}$. According to the rules of the game, the result of comparing *rock* with *scissors*, for example, is that *rock* wins. We therefore give $p : S \times S \to R$ as the matrix:

	rock	*paper*	*scissors*
rock	1	0	1
paper	1	1	0
scissors	0	1	1

where we interpret the row entries as elements of S, column entries as elements of T. Notice how the intransitivity of the rock-paper-scissors game is captured in this matrix, even though R is transitive. □

Observe that, at this stage, it is not clear what the goal of such a problem could be. The function p only tells us how two individuals compare when they interact. We will assume throughout this paper that the goal is to find "the best" elements of S. For instance, if S contains possible chess-playing strategies, and p represents what happens when two such strategies are played against one another, then the goal of such a problem might be to "find the best chess-playing strategies." A clear definition of "the best

strategies" is given in section 3.1.

Our formalization is motived by recent work in *Pareto coevolution*. Pareto coevolution has been used to coevolve cellular automata rules [5] and poker-playing strategy [9]. It has also arisen in the abstract study of problem decomposition; see [14]. The key idea behind Pareto coevolution, articulated in [4], is to treat members of a population as objectives; then the task of a coevolving individual is to remain non-dominated with respect to these objectives. Our framework models these ideas in a formal way.

2 Mathematical Preliminaries

We will make extensive use of concepts and notation from the theory of orders. In this section, we establish notation and recall important definitions. We first define orders as mathematical objects; we then examine some ways these objects combine and relate. For an elementary introduction to these concepts, see [12]; for more in-depth information on concepts like pullbacks, see [2].

2.1 Orders

Recall the *Cartesian product* of two sets S and T is the set of ordered pairs $S \times T = \{(s,t) \mid s \in S, t \in T\}$. A *binary relation* on a set S is a subset $R \subset S \times S$. Given $s_{1,}s_2 \in S$ and a binary relation R on S, we say s_1 and s_2 *relate under* R, written $s_1 R s_2$, when $(s_1,s_2) \in R$. We also say that two elements s_1 and s_2 which relate under R are *comparable* according to R; otherwise, they are *incomparable.*

A binary relation R on a set S is *reflexive* when, for all $s \in S$, sRs. For all $s_1,s_2,s_3 \in S$, if $s_1 R s_2$ and $s_2 R s_3$ imply $s_1 R s_3$, then R is *transitive*. R is *anti-symmetric* if for all $s_1,s_2 \in S$, $s_1 R s_2$ and $s_2 R s_1$ imply $s_1 = s_2$.

Definition 2.1 (Order) A binary relation R is a *preorder* if it is both reflexive and transitive. If a preorder is also anti-symmetric, it is a *partial order*. If, finally, all pairs of individuals from S are comparable according to the partial order R, then R is a *total order* or *linear order*. Note that, in analogy with partial functions, a partial (or pre-) order need not define relations between all pairs of members of S, whereas a total order must. We will call R simply an *order* when it is a pre-, partial, or total order.

A binary relation on a set S expresses the same information as a directed graph with nodes S; the elements of R correspond to the arrows of the graph. Moreover, we can think about graphs in terms of their incidence matrices. Consequently, one can think of these concepts in any of these ways, as convenient.

Now that we know what an order is, we can talk about how to combine two of them together:

Definition 2.2 (Cartesian Product of Preorders) Let S and T be preorders. As sets, $\leq_S \times \leq_T \subset S \times S \times T \times T$. Hence, we can interpret $\leq_S \times \leq_T$ as a relation between $S \times S$ and $T \times T$, relating ordered pairs on S to ordered pairs on T. $\leq_S \times \leq_T$ will be an order of some kind, but as shown in the example below, the type of order may change. Thus, to be precise we define the Cartesian product of two preorders $(S, \leq_S)$ and $(T, \leq_T)$ by $(S, \leq_S) \times (T, \leq_T) = (S \times T, \leq_S \times \leq_T)$. We will write this product simply as $S \times T$. If we take the Cartesian product of a preorder S with itself, we write it as S^2 and the relation in particular as $\leq_S^2$. We define S^n and $\leq_S^n$ similarly.

Example 2.3 The set of real numbers $\mathbb{R}$ is totally ordered by the usual order $\leq$. $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$ is the familiar Cartesian plane. The order on $\mathbb{R}^2$ is $\leq \times \leq = \leq^2$. Unrolling the definition, we arrive at the following relation on $\mathbb{R}^2$: $(x_1,y_1) \leq^2 (x_2,y_2) \Leftrightarrow x_1 \leq y_1 \wedge x_2 \leq y_2$. It is straightforward to verify $\leq^2$ is a partial order. It is not a total order because, for example, $(0,1)$ and $(1,0)$ are incomparable with respect to $\leq^2$. $\square$

Finally, we can also talk about maximal elements in any preorder:

Definition 2.4 (Maximal Elements in Preorders) A *maximum* is an element $\bar{s}$ such that for all $s \in S, s \leq \bar{s}$. A *maximal element* of the preorder S is any element $\hat{s} \in S$ with the property that, for all other $s \in S$, $\hat{s} \leq s \Rightarrow s \leq \hat{s}$. We will write $\widehat{S}$ for the set of all maximal elements of the preorder S. It is possible $S = \emptyset$; consider $\mathbb{R}^2$, for example.

2.2 Functions into Preorders

Let S and T be preorders, and let $f : S \to T$ be a function. f is *monotone*, or *monotonic*, if $s_1 \leq_S s_2 \Rightarrow f(s_1) \leq_T f(s_2)$ for all $s_1,s_2 \in S$. The intuition behind this definition is that f preserves order; put differently, passage through the function f does not destroy any pairwise relations. If we regard S and T as graphs, then a monotone f is exactly a graph homomorphism. f is an *isomorphism* of preorders if f is a monotone bijection and f^{-1} is also monotone. Two isomorphic preorders are "the same;" that is, they typify the same order structure, possibly differing in how their elements are labelled. We will write $S \cong T$ to indicate S and T are isomorphic preorders. Isomorphic preorders have "the same" maximal elements; i.e., if $f : S \to T$ is an isomorphism of preorders, then $f(\widehat{S}) = \widehat{T}$.

An important operation which we will abuse often is that of *pullback* [2]. The idea is that whenever one has a function from a set into a structure like a preorder, it is often possible to "pull" the structure back to the domain set "through" the function. In our case, we often have functions from sets into preorders, and so are able to pull the order structure

from the range back into the domain. Formally,

Definition 2.5 (Pullback Orders) Let S be a set, R a preorder, and let $f : S \to R$ be a function; we will call f a *function into the preorder R*. Given such a function f, we can *pullback* the order of R into S [2]. To be more precise, define a preorder on S, which we write $\leq_f$, as follows: $s_1 \leq_f s_2 \Leftrightarrow f(s_1) \leq_R f(s_2)$ for all $s_1, s_2 \in S$. We will write the resulting preorder $(S, \leq_f)$ as S_f; we will refer to it as the *preorder induced on S by f* . As defined, $\leq_f$ is the largest preorder on S making the function f monotone.

Following the convention in domain theory (e.g., [1]), write $[S \to R]$ for the set of all functions from S to R. If R is a preorder, we can order $[S \to R]$ in two ways. First, we consider the pointwise order:

Definition 2.6 (Pointwise Order) Two functions $f, g \in [S \to R]$ lie in order pointwise, which we write $f \leq_{pw} g$ or just $f \leq g$, if for all $s \in S$, $f(s) \not\geq_R g(s)$. In other words, whenever $f(s)$ and $g(s)$ are related, it must be that $f(s) \leq_R g(s)$. The pointwise order is the default order on $[S \to R]$; when we speak of $[S \to R]$ as if it were ordered, we assume it has the pointwise order.

Example 2.7 Let $S = \{1, \ldots, n\}$, $R = \mathbb{R}$. Then $[S \to \mathbb{R}] = [\{1, \ldots, n\} \to \mathbb{R}] \cong \mathbb{R}^n$ under the isomorphism $f(g) = (g(1), g(2), \ldots, g(n))$, for any $g \in [\{1, \ldots, n\} \to \mathbb{R}]$. The isomorphism expresses the fact that even though they appear different on the surface, $[\{1, \ldots, n\} \to \mathbb{R}]$ and $\mathbb{R}^n$ are expressing the same order structure. □

The second order we consider on $[S \to R]$ is via suborder:

Definition 2.8 (Suborder Order) Recall that an element $f \in [S \to R]$ corresponds to a preorder on S, namely the pullback order S_f defined above. Given two functions f and g, we can ask whether $S_f \subseteq S_g$. Therefore, we write $f \subseteq g$ when $S_f \subseteq S_g$. Explicitly, $f \subseteq g$ holds when, for all $s_1, s_2 \in S$, $f(s_1) \leq_R f(s_2) \Rightarrow g(s_1) \leq_R g(s_2)$.

Example 2.9 *$\leq_{pw}$ and $\subseteq$ are distinct orders on $[S \to R]$.*

Let $S = \{a, b\}$, $R = \{0 < 1 < 2\}$. In Table 1 we give three elements of $[S \to R]$ such that , $f_1 \leq_{pw} f_2$, but $f_1 \not\subseteq f_2$; and, $f_2 \subseteq f_3$, but $f_2 \not\leq_{pw} f_3$. □

	f_1	f_2	f_3
a	0	2	1
b	1	1	0

Table 1: $\leq_{pw}$ and $\subseteq$ can be distinct orders on $[S \to R]$.

3 The Statics of Coevolution

We are now in a position to describe our framework. First, here are some illustrative, running examples:

Example 3.1 *Coevolving sorting networks* [7]

Hillis' seminal paper contains an example when $S \neq T$. Ignoring the details of Hillis' representation and algorithm, $S = \{16\text{-}\mathit{line\ sorting\ networks}\}$ and $T = \{0,1\}^{16}$. R is $\{0 < 1\}$, and the function p expresses whether a sorting network sorts a given test case correctly. □

Example 3.2 *Genetic Algorithm (GA) function optimization*

In many GA problems, we are given a function $f : S \to \mathbb{R}$, and our task is to find one or more elements in S which maximize f. If we use the information in f as an objective function, we can compare two individuals $s_1, s_2 \in S$ via their difference in fitness $f(s_1) - f(s_2)$. Consequently, it is reasonable to define a comparison function $p : S \times S \to \mathbb{R}$ by the formula $p(s_1, s_2) = f(s_1) - f(s_2)$. □

Example 3.3 *Multiobjective Optimization (MOO)* [6]

Similarly, in many MOO problems, we are given a set of objective functions $f_i : S \to \mathbb{R}$, for i in some fixed range $1 \leq i \leq n$. We can combine these objectives into one function $f = \langle f_1, f_2, \ldots, f_n \rangle : S \to \mathbb{R}^n$, which is defined by $\langle f_1, f_2, \ldots, f_n \rangle (s) = (f_1(s), \ldots, f_n(s))$. As we observed in section 2.2, $\mathbb{R}^n$ is a partial order. Furthermore, we know how to subtract in $\mathbb{R}^n$ using pointwise subtraction: $(a_1, \ldots, a_n) - (b_1, \ldots, b_n) = (a_1 - b_1, \ldots, a_n - b_n)$. Therefore, we can proceed as we did in the GA example and define $p : S \times S \to \mathbb{R}^n$ by $p(s_1, s_2) = f(s_1) - f(s_2) = (f_1(s_1) - f_1(s_2), \ldots, f_n(s_1) - f_n(s_2))$. □

3.1 Solution As Set of Maximal Candidates

Generalizing from GA function optimization and MOO, we define a solution concept for coevolutionary problems. The following proposition suggests a connection:

Proposition 3.4 (MOO as Maximization) *The Pareto non-dominated front of a set of objectives $f_i : S \to \mathbb{R}$ $(1 \leq i \leq n)$ is $\widehat{S}_{\langle f_1, \ldots, f_n \rangle}$, the set of maximal elements of the preorder induced on S by the function $\langle f_1, \ldots, f_n \rangle$ into the partial order $\mathbb{R}^n$.*

Proof The Pareto non-dominated front consists of those $\widehat{s} \in S$ which are not dominated by any other $s \in S$. Define the preorder $\preceq$ on S as follows: $s \preceq s' \Leftrightarrow \forall i,\ f_i(s) \leq f_i(s')$ for all $s, s' \in S$. $s \preceq s'$ expresses that s' is not dominated by s. Observe that $s \preceq s' \Leftrightarrow s \leq_{\langle f_1, \ldots, f_n \rangle} s'$ (see section 2.5). The non-dominated front is then $F = \{\widehat{s} \in S \mid \forall s \in S,\ s \preceq \widehat{s}\}$.

Notice that the condition $\forall s \in S, s \preceq \widehat{s}$ is logically equivalent to the condition $\forall s \in S, \widehat{s} \preceq s \Rightarrow s \preceq \widehat{s}$. Consequently, we have that $F = \{\widehat{s} \in S \mid \forall s \in S, \widehat{s} \preceq s \Rightarrow s \preceq \widehat{s}\} = \{\widehat{s} \in S \mid \forall s \in S, \widehat{s} \leq_f s \Rightarrow s \leq_f \widehat{s}\}$, where $f = \langle f_1, \ldots, f_n \rangle$. However, the last set in the chain of equalities is $\widehat{S}_f$, whence we have shown $F = \widehat{S}_f$. □

The importance of proposition 3.4 is that it shows how we can think about MOO problems as maximization problems. In that respect, MOO problems are a generalization of the maximization problems attacked with GAs. It is fair, then, to think of a GA function optimization problem as a MOO problem with a single objective. While this statement is intuitively clear, proposition 3.4 formalizes the intuition.

With a bit of work, we can continue the process of generalization and also view coevolution problems as maximization problems. Start with a coevolution problem expressed with a function $p : S \times T \to R$, and curry this function on T to produce a function $\lambda t.p : S \to [T \to R]$. As we observed, we can preorder $[T \to R]$. Proposition 3.4 suggests the order we should use is $\leq_{pw}$. Consequently, $\lambda t.p$ is a function into a preorder, $[T \to R]$, meaning we can pull the order back to S as in definition 2.5. We now have a preorder on S, the set of candidate solutions. We propose the set of maximal elements of this preorder as a solution to the problem $p : S \times T \to R$. Formally,

Definition 3.5 (Maximal Candidates) The set of maximal candidates of the coevolution problem $p : S \times T \to R$ is $\mathsf{S}_p = \widehat{S}_{\lambda t.p}$. Explicitly, $\mathsf{S}_p = \{s \in S \mid \forall s' \in S, [\forall t \in T, p(s,t) \not\geq_R p(s',t)] \Rightarrow s' = s\}$. We will call $\mathsf{S}_p \subseteq S$ a *solution set* of the problem p.

Here are some examples illustrating the definition:

Example 3.6 *Rock-paper-scissors, revisited*

The rock-paper-scissors incidence matrix is given in example 1.1. Then the functions $\lambda t.p(s)$ are just the rows of the matrix. Comparing these rows pointwise, we see they are all incomparable. Consequently, in rock-paper-scissors, $\mathsf{S}_p = \{rock, paper, scissors\} = S$. □

Example 3.7 *GA and MOO, revisited*

Consider GA optimization or MOO problems where we have objectives $f_i : S \to \mathbb{R}$ for $1 \leq i \leq n$. Treating the objectives "objectively" yields the comparison $p(s,s') = f(s) - f(s')$, where $f = \langle f_1, \ldots, f_n \rangle$. Now observe that in $\mathbb{R}^n$, $f(s_1) - f(s') \leq^n f(s_2) - f(s') \Leftrightarrow f(s_1) \leq^n f(s_2)$, for all $s_1, s_2, s' \in S$, simply by adding $f(s')$ to both sides of the inequality. It follows, therefore, that $\widehat{s}$ is a maximal element with respect to $S_{\lambda t.p}$ if and only if $\widehat{s}$ is non-dominated. As a result, the solution set S_p is exactly the non-dominated front of S (see proposition 3.4). □

In light of the observation that definition 3.5 generalizes common solution concepts used in MOO and GA, it is a natural notion of solution for coevolution as well. Observe that this particular solution concept is distinct from other, common ones, such as "maximize average fitness."

3.2 Test Sets

In this section we consider preorders on the test set T. One possibility is to value tests which produce many distinctions among candidates [5]. A good test is one which can tell us which candidates are better than other candidates. We will formalize this intuition of ideal test set in definition 3.9 and justify it by showing, in theorem 3.10 that the ideal test set induces the same set of maximal candidates as the full test set T.

The suborder order on $[S \to T]$ almost captures what we seek. If $\leq_1$ and $\leq_2$ are two orders on a set S such that $\leq_1 \subseteq \leq_2$, then $\leq_2$ gives the same relations on S that $\leq_1$ does, plus possibly more. However, there is the flaw that the suborder order has a trivial maximum, namely equality. In other words, the trivial relation $R = S \times S$ which says that all elements of S are equal to all others, is such that $\leq \subseteq R$ for any other order $\leq$. More generally, any order which "adds equalities" looks better according to $\subseteq$, even though for our purposes it tells us less information because it reveals fewer distinctions.

In order to repair this problem, we will define an informativeness order $\preceq$. First, let $\leq$ be an order on a set S. Define the set $\leq^{=} = \{(s,s') \mid s, s' \in S, s \leq s' \wedge s' \leq s\}$. Recall that in a partial order, $(s,s') \in \leq^{=} \Rightarrow s = s'$, but this need not be the case in a preorder. The set $\leq^{=}$ tells us which elements in S look "equal" according to $\leq$. We can now use this notion to define a new relation among the orders on S. Roughly speaking, to be informative, an order should have neither incomparable elements nor equal elements. Formally,

Definition 3.8 (Informativeness) Let $\leq_1$ and $\leq_2$ be two orders on S. Say $\leq_2$ is *more informative than* $\leq_1$, written $\leq_1 \preceq \leq_2$, if $\leq_1 \subseteq \leq_2$ and $\leq_2^{=} \subseteq \leq_1^{=}$. If we write $\leq_1 \subseteq^{=} \leq_2$ for the latter condition, we see $\preceq = \subseteq \times \subseteq^{=}$.

We can use $\preceq$ to order $[S \to R]$. Given $f, g \in [S \to R]$, write $f \preceq g$ when $S_f \preceq S_g$; i.e., when the order induced on S by g is more informative than the order induced on S by f.

Now we are in a position to describe the ideal test set. In words, it is the set of maximal elements in T with respect to the pullback of the informativeness order on $[S \to R]$. Formally,

Definition 3.9 (Ideal Test Set) Let $p : S \times T \to R$ be a coevolution problem, and let $\lambda s.p : T \to [S \to R]$ be the curried form of p. Let $[S \to R]$ have the informativeness order $\preceq$. Pull this order back through $\lambda s.p$ into T, and write the resulting order on T as $T_{\preceq}$. Then the ideal test set for this problem is $\mathsf{T}_p = \widehat{T_{\preceq}} \subseteq T$, the set of maximally-informative tests.

That definition 3.9 is useful is borne out by the following:

Theorem 3.10 *Let $p : S \times T \to R$ be a coevolution problem, and consider $p|_{\mathsf{T}_p}: S \times \mathsf{T}_p \to R$, the restriction of p to the maximally-informative tests. For brevity, write q for $p|_{\mathsf{T}_p}$. Then $S_p \cong S_q$; in other words, the maximally-informative set of tests induces the same order on S as the full set of tests T. Consequently, it also induces the same set of maximal candidates.*

Proof See Appendix A. □

Here are some examples:

Example 3.11 *Rock-paper-scissors, revision 3*

In the rock-paper-scissors incidence matrix (see example 1.1), the columns are the $\lambda s.p(t)$. Reading left to right, the induced orders are $\{scissors < rock = paper\}$, $\{rock < paper = scissors\}$, and $\{paper < rock = scissors\}$. None of these orders is a suborder of another; it follows that $\mathsf{T}_p = \{rock, paper, scissors\} = T$. □

Example 3.12 Consider the formal game where $S = T = \{a,b,c\}$, $R = \{0 < 1\}$, and p is given by the matrix

	a	b	c
a	0	1	0
b	1	0	0
c	1	1	1

The orders induced on S are, left to right, $\{a < b = c\}$, $\{b < a = c\}$, and $\{a = b < c\}$. None of these is a suborder of another, so $\mathsf{T}_p = \{a,b,c\} = T$. Notice that $\mathsf{S}_p = \{c\}$, so this example shows S_p and T_p can be distinct; i.e., solutions need not make good tests.

Theorem 3.10 shows that we do not need to use the full set of tests T in order to distinguish individuals in S. In fact, the ideal test set T_p will induce the same order on S and so the same maximal candidates. If T_p is a strict subset of T, then we can solve the same problem p using fewer tests.

Call the cardinality of T_p the *dimension* of the problem p. Then we have the following:

Theorem 3.13 *GA optimization and MOO problems are 1-dimensional when the objectives are treated as in example 3.3*

Proof Let $n \geq 1$ and $f_i : S \to \mathbb{R}$ for $1 \leq i \leq n$. The observation in example 3.7 that $f_i(s_1) - f(s') \leq f(s_2) - f(s') \Leftrightarrow f(s_1) \leq f(s_2)$ leads to the result, because all tests s' are equivalent (i.e., are in $\preceq^{=}$). □

Remark A MOO problem with n objectives looks like it should be n-dimensional. However, in theorem 3.13 we are using the objectives to compare pairs of individuals. Then the *individuals* are the tests, not the objectives. In that case, any individual will do as a test, making the MOO problem 1-dimensional. If we were to treat the objectives themselves as tests, then a MOO problem with n objectives would be n-dimensional.

The content of theorem 3.13 is that "difficult" coevolution problems appear to have dimension > 1. Rock-paper-scissors is 3-dimensional, for example. By inspecting the structure of the tests for a problem p, we are able to say whether the problem is coevolutionary, independently of any search algorithms we might employ. In other words, theorem 3.13 is a tool for categorizing problems. Algorithm choices might make the fitness of an individual dependent on the constituents of the current population, which is clearly coevolutionary. Nevertheless, we are able to distinguish problems on the basis of their test set structure alone, prior to any algorithm choices. Problems with simple test set structure are likely to be simpler to solve in practice; see, for instance, Juillé's discussion of the cellular automaton majority function problem in [8].

4 Discussion: Coevolution Issues

We conclude by examining three common coevolution issues from the perspective offered by coevolutionary statics.

In order to discuss well-known issues which arise in coevolution, we will need to reconsider coevolutionary dynamics. To remain within the content of coevolutionary statics, we will consider a fixed point in time during coevolutionary search. If the overarching problem is $p : S \times T \to R$, then what we have been able to examine is $p|_{S' \times T'}: S' \times T' \to R$, where $S' \subseteq S$ and $T' \subseteq T$. In other words, we assume that up to this point in search, we have encountered the candidates in S' and the tests in T', and have been able to assess the value of p for all pairs in $S' \times T'$. The task of an algorithm is to use this information to decide how to update S' and T' in such a way that we ultimately find S_p or some subset thereof.

The problem of *collusion* (e.g., [10]) occurs when S' and T' are updated in a way which increases the apparent payoff for both candidates and tests, but does not move S' closer to S_p. Collusion might occur if the algorithm implicitly treats the function $p : S \times T \to R$ objectively. By "objectively" we mean the algorithm searches for pairs (s,t) which maximize p. Clearly, updating in this way is prob-

lematic, because a pair (s,t) with an easy, uninformative test t will have a high value for p, regardless of where s lies with respect to S_p. Nevertheless, "naive coevolution" which gives an individual fitness based on its average score against a population, often implicitly favors colluding pairs; see, for instance, the discussion of the meta-game of learning in [10].

The *Red Queen Effect* [3] is related to collusion. The essence of the Red Queen Effect is that we are unable to tell the difference between collusion and true progress on the basis of fitness values alone. The reason is essentially that $p(s,t) = p(s',t')$ could occur for two reasons: s is better than s' and t is a harder test than t'; or, s and s' are equals, as are the tests t and t'. Values of p alone do not allow us to see progress. Coevolutionary statics offers some hope in this case. As we have emphasized, the important function to consider is the curried form of p, $\lambda t.p$, in particular the order $S_{\lambda t.p}$ which this function induces on the candidate set S. The "goal" of coevolution is then to climb that order to arrive at the maximal elements S_p. To track progress, we must make observations which allow us to see if the algorithm really is climbing $S_{\lambda t.p}$. Even better, we should arrange our algorithms to guarantee, as much as possible, progress up the order $S_{\lambda t.p}$. Pareto coevolution involves heuristics for achieving this goal.

Finally, *focusing* [15] refers to the ability of coevolving opponents to challenge one another by testing weak dimensions of performance. An issue which arises in this context is *overspecialization*. In the language of our framework, T' is well-focused when the tests in T' are informative, meaning they indicate many discriminations among the candidates in S'. Otherwise, we say that S' and T' are *disengaged*. Heuristically, it appears that if we update the tests in T' so that they become more informative relative to the candidates S', then we minimize the risk that the candidates and tests will become disengaged.

Acknowledgements

The authors wish to thank the members of the DEMO lab for their support and encouragement. In particular, this paper benefited inestimably from long conversations with Sevan Ficici, Edwin de Jong and Richard Watson. We also wish to thanks the reviewers for their useful comments.

A Appendix

Proof of theorem 3.10

Let $p : S \times T \to R$; curry this to $\lambda t.p : S \to [T \to R]$, and consider the restriction $\lambda t.p\,|_{\mathsf{T}_p} : S \to [\mathsf{T}_p \to R]$. The theorem is equivalent to showing $\lambda t.p(S) \cong \lambda t.p\,|_{\mathsf{T}_p}(S)$. This isomorphism is identical to the equivalence $\forall s_1, s_2 \in S, s_1 \leq_{\lambda t.p} s_2 \Leftrightarrow s_1 \leq_{\lambda t.p|_{\mathsf{T}_p}} s_2$. This equivalence is in turn equivalent to $\lambda t.p(s_1) \leq_{pw} \lambda t.p(s_2) \Leftrightarrow \lambda t.p\,|_{\mathsf{T}_p}(s_1) \leq_{pw} \lambda t.p\,|_{\mathsf{T}_p}(s_2)$, by definition of $\leq_f$. We finally have the equivalence with $\forall t \in T, p(s_1,t) \not\geq_R p(s_2,t) \Leftrightarrow \forall t \in \mathsf{T}_p, p(s_1,t) \not\geq_R p(s_2,t)$ by definition of $\leq_{pw}$. To sum up, proving theorem 3.10 is equivalent to proving this last equivalence.

The forward implication holds trivially, because $\mathsf{T}_p \subseteq T$. Consequently, we focus our attention on showing $\forall t \in \mathsf{T}_p, p(s_1,t) \not\geq_R p(s_2,t) \Rightarrow \forall t \in T, p(s_1,t) \not\geq_R p(s_2,t)$, for all $s_1, s_2 \in S$. If we can show this implication, we have the result. So, let $t \in T$. By definition of T_p, $\exists \widehat{t} \in \mathsf{T}_p$ such that $t \preceq \widehat{t}$. In particular, $t \subseteq \widehat{t}$. Assume $p(s_1,t) \geq_R p(s_2,t))$; then it follows $p(s_1,\widehat{t}) \geq_R p(s_2,\widehat{t},$ because $\widehat{t}$ is more informative than t. This is a contradiction; thus, it must be that $p(s_1,t) \not\geq_R p(s_2,t)$. The latter holds for any $t \in T$; therefore, we have our result. □

References

[1] Samson Abramsky and Achim Jung. Domain theory. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science Volume 3*, pages 1–168. Oxford University Press, 1994.

[2] Michael Barr and Charles Wells. *Category Theory for Computing Science*. Prentice Hall International Series in Computer Science. Prentice Hall, New York, 1st edition, 1990.

[3] Dave Cliff and Geoffrey F. Miller. Tracking the red queen: Measurements of adaptive progress in coevolutionary simulations. In *European Conference on Artificial Life*, pages 200–218, 1995.

[4] S. G. Ficici and J. B. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In Hans-Paul Schwefel Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, editor, *Parallel Problem Solving from Nature - PPSN VI 6th International Conference*, Paris, France, 16-20 2000. Springer Verlag.

[5] Sevan G. Ficici and Jordan B. Pollack. Pareto optimality in coevolutionary learning. In *European Conference on Artificial Life*, pages 316–325, 2001.

[6] Carlos M. Fonseca and Peter J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[7] W. Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In Christopher G. Langton, Charles Taylor, J. Doyne Farmer, and Steen Rasmussen, editors, *Artificial Life*

II, volume X, pages 313–324. Addison-Wesley, Santa Fe Institute, New Mexico, USA, 1990 1992.

[8] Hugues Juille and Jordan B. Pollack. Coevolving the ideal trainer: Application to the discovery of cellular automata rules. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 519–527, University of Wisconsin, Madison, Wisconsin, USA, 22-25 1998. Morgan Kaufmann.

[9] Jason Noble and Richard A. Watson. Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for pareto selection. In L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, pages 493–500, San Francisco, CA, 2001. Morgan Kaufmann Publishers.

[10] Jordan B. Pollack and Alan D. Blair. Co-evolution in the successful learning of backgammon strategy. *Machine Learning*, 32(3):225–240, 1998.

[11] Christopher D. Rosin. *Coevolutionary search among adversaries*. PhD thesis, University of California, San Diego, San Diego, CA, 1997.

[12] Edward R. Scheinerman. *Mathematics: A Discrete Introduction*. Brooks/Cole, Pacific Grove, CA, 1st edition, 2000.

[13] Karl Sims. Evolving virtual creatures. *Computer Graphics*, 28(Annual Conference Series):15–22, 1994.

[14] R. A. Watson and J. B. Pollack. Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In Hans-Paul Schwefel Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, editor, *Parallel Problem Solving from Nature - PPSN VI 6th International Conference*, Paris, France, 16-20 2000. Springer Verlag.

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