Monotonic Solution Concepts in Coevolution

Sevan G. Ficici
Computer Science Department, Brandeis University
Waltham, Massachusetts USA
sevan@cs.brandeis.edu

ABSTRACT

Assume a coevolutionary algorithm capable of storing and utilizing all phenotypes discovered during its operation, for as long as it operates on a problem; that is, assume an algorithm with a monotonically increasing knowledge of the search space. We ask: If such an algorithm were to periodically report, over the course of its operation, the best solution found so far, would the quality of the solution reported by the algorithm improve monotonically over time? To answer this question, we construct a simple preference relation to reason about the goodness of different individual and composite phenotypic behaviors. We then show that whether the solutions reported by the coevolutionary algorithm improve monotonically with respect to this preference relation depends upon the solution concept implemented by the algorithm. We show that the solution concept implemented by the conventional coevolutionary algorithm does not guarantee monotonic improvement; in contrast, the game-theoretic solution concept of Nash equilibrium does guarantee monotonic improvement. Thus, this paper considers 1) whether global and objective metrics of goodness can be applied to coevolutionary problem domains (possibly with open-ended search spaces), and 2) whether coevolutionary algorithms can, in principle, optimize with respect to such metrics and find solutions to games of strategy.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence—*Problem Solving, Control Methods, and Search*; I.2.6 [**Computing Methodologies**]: Artificial Intelligence—*Learning*

General Terms

Theory

Keywords

Coevolution, monotonic progress, solution concepts

GECCO'05, June 25–29, 2005, Washington, DC, USA.

1. INTRODUCTION

Coevolutionary algorithms are stochastic search methods that are frequently used to find (or approximate) solutions to games of strategy. Nevertheless, whether coevolutionary-algorithm dynamics are actually directional and inclined to converge onto global solutions (as opposed to performing mere local adaptation) is a contentious issue [7, 8, 14]. Further, some argue that an objective metric of goodness is an ill-conceived proposition for coevolution; for example, in [13] frequency-dependent fitness (usually found in coevolutionary algorithms) is equated with the lack of an "absolute goal."

In this paper, we investigate how a coevolutionary algorithm may behave if it never discards any information learned during its operation—all strategies discovered during search are retained and utilized. As a general principle, the expected quality of a solution returned by a search method should improve as the search method is given more computational resources in time and/or space. More desirable still, if we repeatedly query a search method (at appropriate points during its execution) on any single run, then the quality of the solution returned by the method should improve monotonically—that is, the quality of the solution at time $t+1$ should be no worse than the quality at time t.

Given a coevolutionary algorithm that does not discard information during execution, can we expect it to meet the *desideratum* of monotonicity? On the one hand, we may expect that it must, since the algorithm's knowledge of the domain is ever increasing; the accumulation of strategies makes the evaluation process increasingly comprehensive and provides growing genetic diversity for the variation operators. On the other hand, we may remind ourselves that many of the problems associated with coevolution arise because evaluation is never fully comprehensive, and therefore can be misled when the domain contains intransitivities; new information may drastically shift our perspective. How, then, can we guarantee that the solution will improve monotonically with time (in some global, objective sense)?

This paper shows that, when information is not discarded, a coevolutionary algorithm may or may not behave monotonically, depending upon the *solution concept* used by the algorithm. We develop a formal framework to discuss solution concepts and monotonicity and show that certain solution concepts (e.g., Nash equilibrium) guarantee monotonicity and others (including that most commonly used in coevolution) do not. Clearly, real-world algorithms have finite capacity and must occasionally discard information; nevertheless, a monotonic improvement in solution quality over

time can be approximated by a finite-memory system if the solution concept permits (see for example [10], which uses the Nash-equilibrium solution concept).

We proceed as follows. Section 2 reviews foundational issues regarding solution concepts in general and how solutions are commonly expressed in coevolutionary algorithms. Section 3 introduces the basic notions of our formal framework. Section 4 defines and discusses a preference relation between candidate solutions that forms the basis of our reasoning about monotonic solution concepts. Section 5 examines two solution concepts from the perspective of monotonicity: the one conventionally implemented in coevolutionary algorithms and also Nash equilibrium. Sections 6–8 provide further discussion, examine the relevance of our results to open-ended domains, and offer concluding remarks. This paper reports work originally presented in [9].

2. FOUNDATIONS

2.1 Solution Concepts

Fundamental to all search problems is the notion of a *solution concept*. Whatever properties our problem domain may possess, and however we embed that domain into a search space, we require a solution concept to indicate which locations in the search space—if any—constitute *solutions* to our problem. A solution concept thus partitions a search space into two classes: solutions and non-solutions. Typically, the two classes are distinguished in a systematic way—by some number of measurable properties that are present or absent in class members; in general, however, any arbitrary binary partition constitutes a valid solution concept. Thus, a search space can have an exponential number of solution concepts applied to it. When we apply a particular solution concept to a search space, we obtain a particular *search problem*.

While solution concepts are intrinsic to search problems, they must nevertheless be implemented by *search algorithms*. We can choose from any number of algorithms to solve a given search problem. Some algorithms may be more or less efficient than others with respect to our search problem, yet all of the algorithms must implement the same solution concept to be *consistent* with our search problem; algorithms that implement different solution concepts solve different search problems, by definition (even when used on the same search space). Solution concepts thus form the nexus of search problems and processes to solve them, and are therefore of very great importance.

We belabor our point because a coevolutionary algorithm's fitness function, which determines an individual's likelihood of survival, is often conflated with the solution concept, and this leads to difficulties. Recognition of this fusion is crucial to the improvement of coevolutionary algorithm design [9].

2.2 Expression of a Solution

When we apply evolution to a static single-objective problem, the most fit individual of the final population is considered the "solution"; if more than one individual in the final population achieves this maximal fitness, then we are indifferent as to which is selected (even if these individuals are genetically or phenotypically distinct). When we apply evolution to a static multi-objective problem, then the solution that is delivered is typically the *Pareto front*, which is a *set* of non-dominated feasible members of a trade-off surface [11]; these individuals are either in the evolving population or in an archive of some sort. The practice of coevolution follows, for the most part, the convention that the solution to a coevolutionary search problem will be a single "best" individual (or possibly a set of individuals where multiple populations that coevolve with each other each contribute their best individual, e.g., [24]). Indeed, the efficacy of some coevolutionary algorithms and the correctness of some analyses are predicated on the existence of a single, globally perfect individual behavior (e.g., [26, 27]).

In [23] experiments on robotic pursuit-and-evasion lead to the observation that the notion of a single global winner may be inadequate. Other research explicitly embraces the belief that a solution may need to be expressed as a *collection* of evolved agents (all from the same population), where no single agent in the collection can be considered a proper solution on its own. (In the case of multi-population coevolution, each population may need to contribute such a collection.) For example, in [4] a collection of strategies for the Iterated Prisoner's Dilemma (IPD) is evolved with a speciating evolutionary algorithm. A novel gating mechanism is used (once evolution is complete) to determine which strategy in the collection likely offers the most effective response against a particular opponent at every point during a 100-iteration game; this procedure is shown to improve robustness. We note that the "output" of this framework is not entirely evolved: the gating mechanism is hand-built. Nevertheless, the solution obtained is clearly not in the form of a single best individual.

Finally, evolutionary game theory [22] regards the state of the entire population as a solution (provided the population-state is a stable fixed-point of the replicator dynamics). For example, in the Hawk-Dove game, the Nash equilibrium solution is obtained when the population reaches a particular ratio of Hawks and Doves; this is known as a *polymorphism*. Thus, the solution here is not a single individual (e.g., a Hawk), nor is it a collection of unique individuals (e.g., a Hawk and a Dove); rather, the solution is a specific *distribution* over the two strategy types.

In this paper, we use a construct called a *behavior complex* to represent various types of strategy collections (including singletons). The space of complexes forms our search space.

3. FORMALISM

3.1 Symmetric Domains

For brevity and ease of exposition, we limit our analyses to symmetric games for two players. Familiar examples of two-player domains include zero-sum board games such as Tic-Tac-Toe or Backgammon. These domains are asymmetric because they distinguish the roles of the two players: one player moves first. As a result, each player has available to it a unique set of behaviors, or *strategies*, for use in the game. Examples of symmetric domains are games such as Rock-Paper-Scissors, various "numbers games" [31], and the IPD. In these games, the player roles are not distinguished; both players share the same set of possible behaviors.

Let $\mathfrak{D}$ denote the domain of interest. Let $\mathcal{S}_P$ be the set of *pure strategies* that the domain makes available to both players. We may think of $\mathcal{S}_P$ as a set of "basis behaviors." In addition to the pure strategies, the domain makes available to the players an uncountable infinity of *mixed strategies*; these are stochastic behaviors based upon probability distributions over the pure strategies. Let $\mathcal{S}_M$ be the set of

mixed strategies. Note that pure strategies are degenerate mixtures, where the entire probability mass is on a single pure strategy.

When the two players interact, each player receives a *payoff* that generally depends upon both the player's own strategy and the strategy used by the other player. Let G be a look-up function (which we term a *game*) such that, when Players 1 and 2 use strategies s_α and s_β (where $s_\alpha, s_\beta \in \mathcal{S}_M$ and s_α may equal s_β), respectively, the expected payoffs to Players 1 and 2 are $G(s_\alpha, s_\beta)$ and $G(s_\beta, s_\alpha)$, respectively. (The expected payoffs of mixed strategies are weighted sums of pure-strategy payoffs, where the weights correspond to the probability distributions of the mixtures.)

3.2 Evolutionary Substrate

Our search of the domain $\mathfrak{D}$ is mediated by an *evolutionary substrate* and associated variation operators. In EA parlance, the substrate is a *genome*, and any instantiation of the genome is a *genotype*; the interpretation of a genotype results in a *phenotype*, which in-turn yields some behavior that we observe and measure. Examples of evolutionary substrates include grammars, neural networks, genetic programs, and bit-strings.

Let $\mathcal{P}$ be the universe of phenotypic behaviors enabled by a substrate (via its interpretation). $\mathcal{P}$ may overlap in any number of ways with the universe of behaviors specified by the domain $\mathfrak{D}$ (i.e., $\mathcal{S}_M$). We can imagine five general cases. First, the two sets $\mathcal{P}$ and $\mathfrak{D}$ may be identical; thus, every behavior possible in the domain is generated by the substrate and no behavior generated by the substrate is external to the domain. Second, the set $\mathcal{P}$ may be a proper subset of $\mathfrak{D}$. In this case, all behaviors generated by the substrate belong to the domain, but not all domain behaviors are covered. For example, the genome may allow $\mathcal{P}$ to contain only the pure strategies (or some other subset of the mixed strategies). In particular, the domain behaviors we care about most—those that constitute solutions—may be excluded. Third, the set $\mathcal{P}$ may be a proper superset of $\mathfrak{D}$; in this case, all the behaviors of the domain are covered, but the substrate also generates behaviors that fall outside of the domain. (An example of this situation can be found in [2] where genetic programs are evolved to play Tic-Tac-Toe; in addition to generating all the possible (deterministic) strategies for the domain, their substrate also generates behaviors that make illegal moves. Such behaviors are not disallowed, but rather cause the player making an illegal move to forfeit a turn. The accommodation of these "illegal" behaviors represents, essentially, an extension of the domain—a new rule to the game.) Fourth, neither set may contain the other, but a non-empty intersection may exist; this case combines the features of Cases 2 and 3. Fifth, the two sets may have an empty intersection.

Regardless of the relationship between $\mathcal{P}$ and $\mathfrak{D}$, we must recognize that we are not directly searching the domain of interest, but instead traversing the space of behaviors made available by $\mathcal{P}$. Thus, $\mathcal{P}$ is the *de facto* domain.

3.3 Games and Sub-Games

Let G_U denote the function (or *game*) defined over the universe of behaviors enabled by $\mathcal{P}$ such that $G_U(s_i, s_j)$ and $G_U(s_j, s_i)$ are the expected payoffs to Players 1 and 2 using strategies s_i and s_j, respectively. A game G_α is a *sub-game* of another game G_β, denoted $G_\alpha \subseteq G_\beta$, if and only if the set of phenotypes $\mathcal{P}_\alpha$ over which G_α is defined is a subset of the set of phenotypes $\mathcal{P}_\beta$ over which G_β is defined. One particular sub-game of G_U that we are interested to consider is G_W, which we define to reflect our *state of knowledge*, that is, the set of phenotypes that our coevolutionary algorithm has discovered. We discuss G_W further below. For convenience, we will use a game G_x and its corresponding phenotype set $\mathcal{P}_x$ interchangeably, often treating G_x as a set of phenotypes. We also use capital letters (e.g., A, B) to denote games.

3.4 Behavior Complexes

As we note above, the space of phenotypes made available by the genomic representation may exclude certain behaviors that belong to the domain of interest. In particular, $\mathcal{P}$ may exclude some or all mixed strategies, yet a missing mixed strategy may be a solution to our game. To accommodate this possibility, we introduce the *behavior complex* $\mathcal{C}$. A behavior complex is a collection of phenotypic behaviors that are combined (via a probability distribution, such that members of the complex are played with non-zero probability) to correspond to some mixed strategy $s_m \in \mathcal{S}_M$.

Thus, our search effort ultimately concentrates on the space of complexes. Further, our solution concept does not operate directly upon the problem domain of interest ($\mathfrak{D}$), but rather upon the the *de facto* domain, which is observed and measured via these behavior complexes. A solution is always expressed as a behavior complex (even if the complex contains only a single phenotype).

3.5 Solution Concepts and Solutions

A *solution concept* (or, optimality concept) $\mathcal{O}$ is defined either *extensionally* or *intensionally*. A solution concept can simply partition the space of behavior complexes into solutions and non-solutions without stating any underlying properties by virtue of which a behavior complex is considered a solution; a solution concept so defined is extensional. Alternatively (and usually the case in real-world algorithms), a solution concept will state a number of properties that a behavior complex must possess to be a solution; such a solution concept is intensional.

The assessment of a behavior complex $\mathcal{C}$, to determine whether or not it is a solution, is always performed with respect to some given sub-game $G_\alpha \subseteq G_U$ and corresponding set of phenotypes $\mathcal{P}_\alpha \subseteq \mathcal{P}$, where $\mathcal{C} \subseteq \mathcal{P}_\alpha$. Let $\mathcal{C}^*$ denote a behavior complex that is a solution to sub-game G_α according to solution concept $\mathcal{O}$; let $\mathbb{C}^*(G_\alpha, \mathcal{O})$ be the set of all solutions to G_α according to $\mathcal{O}$.

3.6 State of Knowledge

The phenotypes that 1) have been discovered by a search heuristic and 2) are still in the computer's memory, such that they can be utilized by the heuristic, are defined as the heuristic's *state of knowledge*. We denote the state of knowledge at time t as $\mathcal{W}^t$, where $\mathcal{W}^t \subseteq \mathcal{P}$. At any time-step t that we query the search heuristic, the behavior complex $\mathcal{C}$ returned by the heuristic should be a solution to the game G_W (defined over $\mathcal{W}^t$) according to solution concept $\mathcal{O}$, i.e., $\mathcal{C} \in \mathbb{C}^*(G_W, \mathcal{O})$. Otherwise, the heuristic does not implement $\mathcal{O}$. Note that all strategies included in $\mathcal{C}$ must also be in $\mathcal{W}^t$—we cannot use strategies of which we lack knowledge. When $G_W = G_U$, we expect the search heuristic to return a behavior complex that is a global solution.

In the analyses we perform in Section 5, we assume a coevolutionary algorithm where $\mathcal{W}^t \subseteq \mathcal{W}^{t+1}$ for all t. That is, the algorithm's knowledge of the search space increases monotonically over time. We are interested to know if the quality of the solution reported by the algorithm increases monotonically over time, as well. For convenience, we often use $\mathcal{W}$ and $G_{\mathcal{W}}$ interchangeably.

3.7 From Domain to Solution

Let us review our formalism. The problem domain $\mathfrak{D}$ specifies a universe of behaviors that we are interested to explore. Nevertheless, we are unable to traverse $\mathfrak{D}$ directly. Genotypes are the objects to which selection pressure and variational operators are applied; on a mechanistic level, we perform search in the space of genotypes. The substrate (along with an interpretive process) provides a way to express behaviors (the phenotypes $\mathcal{P}$) and thereby realize our exploration of the domain; the phenotypes provide the behaviors we can actually observe. Individual phenotypes might not constitute solutions according to our solution concept $\mathcal{O}$, thus we use behavior complexes to express the space of behaviors that we actually explore on the conceptual level.

4. PREFERENCE RELATION

Here we define and examine a preference relation over the space of behavior-complexes. The relation forms the basis of our reasoning about monotonic solution concepts.

4.1 Definition

Given some game G_U and solution concept $\mathcal{O}$, we can define the following preference relation (or predicate). We *prefer* one complex $\mathcal{C}_\alpha$ over another complex $\mathcal{C}_\beta$, denoted $\mathcal{C}_\alpha \overset{\mathcal{O}}{\succ} \mathcal{C}_\beta$, if and only if every game for which $\mathcal{C}_\beta$ is a solution is a proper sub-game of some game for which $\mathcal{C}_\alpha$ is a solution; that is, we prefer $\mathcal{C}_\alpha$ because it generalizes $\mathcal{C}_\beta$ (by applying to super-games). When the preference relation does not hold between a pair of complexes $\mathcal{C}_\alpha$ and $\mathcal{C}_\beta$, denoted $\mathcal{C}_\alpha \approx \mathcal{C}_\beta$, then we neither prefer $\mathcal{C}_\alpha$ to $\mathcal{C}_\beta$ nor *vice versa*.

The preference relation implies that we prefer any solution $\mathcal{C}^*$ to G_U over any non-solution $\mathcal{C}$; further, given two or more solutions to G_U, the relation implies that we have no preference for one solution over another. (If we have an informal preference for one solution over another, then we can formalize our preference by *refining* the solution concept, as discussed in [9].)

DEFINITION 1 (PREFERENCE RELATION). *Let $\mathcal{G}_\alpha$ be the set of games for which $\mathcal{C}_\alpha$ is a solution; let $\mathcal{G}_\beta$ be the set of games for which $\mathcal{C}_\beta$ is a solution. $\mathcal{C}_\alpha \succ \mathcal{C}_\beta$ iff $\forall G_\beta \in \mathcal{G}_\beta : \exists G_\alpha \in \mathcal{G}_\alpha : G_\beta \subset G_\alpha \subseteq G_U$.*

4.2 Reflexivity, Symmetry, and Transitivity

The preference relation is not reflexive—we will never prefer a configuration to itself: $\mathcal{C}_\alpha \approx \mathcal{C}_\alpha$. Non-reflexivity follows from the definition of the preference relation; given the set $\mathcal{G}_\alpha$ of games for which $\mathcal{C}_\alpha$ is a solution, there must exist at least one game $G \in \mathcal{G}_\alpha$ that does not have a superset in $\mathcal{G}_\alpha$.

The preference relation is not symmetric—if we prefer one configuration to another, then the converse is not true: $\mathcal{C}_\alpha \succ \mathcal{C}_\beta \Longrightarrow \mathcal{C}_\beta \not\succ \mathcal{C}_\alpha$. Non-symmetry also follows from the definition of the preference relation. Let $\mathcal{G}_\alpha$ be the set of games for which configuration $\mathcal{C}_\alpha$ is a solution and $\mathcal{G}_\beta$ be the set of games for which $\mathcal{C}_\beta$ is a solution. Since we prefer $\mathcal{C}_\alpha$ to $\mathcal{C}_\beta$, each game in $\mathcal{G}_\beta$ is a sub-game of some game in $\mathcal{G}_\alpha$. If we simultaneously prefer $\mathcal{C}_\beta$ to $\mathcal{C}_\alpha$, then each game in $\mathcal{G}_\alpha$ must also be a sub-game of some game in $\mathcal{G}_\beta$; but, for this to be true, the subset relation must allow for an intransitivity, which it does not.

While the preference relation is neither reflexive nor symmetric, it is transitive—if we prefer configuration $\mathcal{C}_\alpha$ to $\mathcal{C}_\beta$, and prefer $\mathcal{C}_\beta$ to $\mathcal{C}_\gamma$, then we prefer $\mathcal{C}_\alpha$ to $\mathcal{C}_\gamma$. The transitivity of the preference relation follows directly from the transitivity of the subset relation, as Figure 1 illustrates. Since each game in $\mathcal{G}_\gamma$ is a subset of some game in $\mathcal{G}_\beta$, and each game in $\mathcal{G}_\beta$ is a subset of some game in $\mathcal{G}_\alpha$, then each game in $\mathcal{G}_\gamma$ must also be a subset of some game in $\mathcal{G}_\alpha$.

Thus, the preference relation yields a partial ordering over the space of behavior complexes.

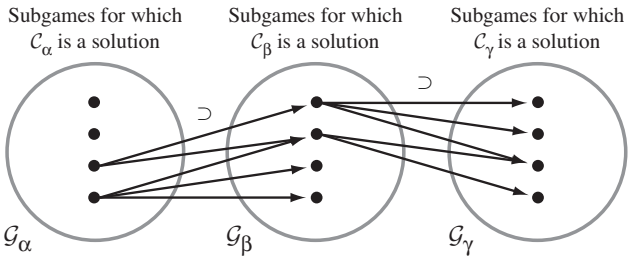

Figure 1: Transitivity of preference relation follows from transitivity of subset relation. Directed edge indicates superset relation between two games.

4.3 Monotonicity

Let $\mathcal{G}_x$ be the set of games for which complex $\mathcal{C}_x$ is a solution and $\mathcal{G}_y$ the set of games for which complex $\mathcal{C}_y$ is a solution. The preference relation says that we prefer $\mathcal{C}_x$ to $\mathcal{C}_y$ if and only if every game in $\mathcal{G}_y$ is a sub-game of some game in $\mathcal{G}_x$. Of course, not every game in $\mathcal{G}_x$ must be a super-game; in particular, the preference relation allows for the existence of a game in $\mathcal{G}_x$ that is a sub-game of a game in $\mathcal{G}_y$, as shown in Figure 2. Here, we prefer $\mathcal{C}_x$ to $\mathcal{C}_y$, yet game $E \in \mathcal{G}_x$ is a sub-game of game $D \in \mathcal{G}_y$.

When a complex $\mathcal{C}$ is a solution for games G_α and G_γ, where $G_\alpha \supset G_\gamma$, but not for some game G_β, where $G_\alpha \supset G_\beta \supset G_\gamma$, then we call the solution concept $\mathcal{O}$ *non-monotonic*. We define a *monotonic* solution concept to be one such that every complex $\mathcal{C}$ that is a solution to games G_α and G_γ, where $G_\alpha \supset G_\gamma$, will also be a solution to any game G_β, where $G_\alpha \supset G_\beta \supset G_\gamma$. The solution concept in Figure 2 is therefore non-monotonic: $\mathcal{C}_x$ is a solution for games C and E, but not of D, and $C \supset D \supset E$. The relationship between game-sets $\mathcal{G}_x$ and $\mathcal{G}_y$ allows a search algorithm to exhibit non-monotonic behavior.

4.4 Dynamics of Non-Monotonic Concepts

For any solution concept $\mathcal{O}$, the preference relation provides a global partial ordering over the space of behavior complexes. Nevertheless, if $\mathcal{O}$ is non-monotonic, then the process of search—even when it does not discard any information discovered during its operation—may be unable to conform to the preference relation and will contradict it. For example, let us return to Figure 2. If the game $E \in \mathcal{G}_x$ is our state of knowledge at time t (i.e., $\mathcal{W}^t = E$), then the solution

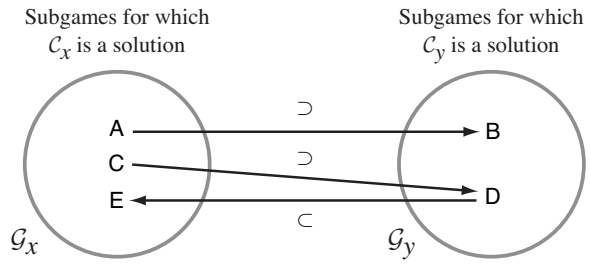

Figure 2: Non-monotonic solution concept.

returned by the search heuristic when queried is the complex $\mathcal{C}_x$. Let us say that at time $t+1$, the heuristic discovers new strategies and our state of knowledge is expanded to include these additional strategies such that our new state of knowledge is game D. Since $D \notin \mathcal{G}_x$ and $D \in \mathcal{G}_y$, our solution is now complex $\mathcal{C}_y$. By discarding $\mathcal{C}_x$ in favor of $\mathcal{C}_y$, the algorithm contradicts the preference relation (which says that we prefer $\mathcal{C}_x$ to $\mathcal{C}_y$). If the state of knowledge is expanded further to become game C, then the search algorithm contradicts itself, in a sense, by re-adopting a solution it had previously discarded.

4.5 Dynamics of Monotonic Concepts

The definition of a monotonic solution concept guarantees that coevolutionary search, when it does not discard information, will always conform to the preference relation. To show that this must be the case, let us imagine two states of knowledge, $\mathcal{W}^t$ and $\mathcal{W}^{t+k}$, where $\mathcal{W}^t \subset \mathcal{W}^{t+k}$ and $k \geq 1$. Further, let us say that the solution reported for $\mathcal{W}^t$ is complex $\mathcal{C}_x$ and for $\mathcal{W}^{t+k}$ is $\mathcal{C}_y$. Now, if the solution concept is monotonic, then we cannot prefer (by Definition 1) $\mathcal{C}_x$ to $\mathcal{C}_y$. If we prefer $\mathcal{C}_x$ to $\mathcal{C}_y$, then there must exist some game G that is a superset of $\mathcal{W}^{t+k}$ for which $\mathcal{C}_x$ is a solution. But, if $\mathcal{C}_x$ is a solution of both games $\mathcal{W}^t$ and G, and $\mathcal{W}^t \subset \mathcal{W}^{t+k} \subset G$, then either $\mathcal{C}_x$ must also be a solution to $\mathcal{W}^{t+k}$ (in which case we do not discard $\mathcal{C}_x$ and transition to $\mathcal{C}_y$), or the solution concept is not monotonic.

We can construct a directed graph of the preference relation where each vertex is a behavior complex, and for each pair of complexes $\mathcal{C}_x$ and $\mathcal{C}_y$, where we prefer $\mathcal{C}_x$ to $\mathcal{C}_y$, there is a directed edge from $\mathcal{C}_x$ to $\mathcal{C}_y$. If search does not discard information, then the property of monotonicity means that once we visit a vertex on the graph, we will never follow an edge leading back to that vertex or to any of its descendants in the graph. Thus, a monotonic solution concept means that the quality of the result returned by a search heuristic (assuming that it does not discard information) will also increase monotonically over time; this is not guaranteed to be true for a non-monotonic solution concept, even when information is never discarded.

5. SOLUTION CONCEPTS

Having discussed the properties of monotonic and non-monotonic solution concepts in general, we now investigate some specific solution concepts. We show that the solution concept commonly implemented in coevolutionary algorithms is not monotonic, whereas the game-theoretic concept of Nash equilibrium is.

5.1 Best-Scoring Strategy

5.1.1 Description

In the conventional single-population coevolutionary algorithm, we constrain the solution to be embodied by a single individual; thus, the space of behavior complexes is restricted to (i.e., the same as) the space of phenotypes. We define the solution to be (the strategy implemented by the phenotype of) the individual in the final population with the highest fitness; when queried during its operation, the algorithm returns the fittest individual of the current population. An individual's fitness is derived from the sum of payoffs obtained by *complete mixing* (interaction with all other members of the population). Thus, fitness is sensitive not only to which strategies are represented in the population, but also the proportions with which each strategy is represented; fitness is *frequency-dependent*. Let us call this solution concept "best-in-the-ecology" (BITE).

To simplify our analysis we consider a special case of the BITE solution concept that is frequency-independent; we will call our alternative solution concept *best-scoring strategy* (BSS). Given the set $\mathcal{S}$ of unique strategies, the BSS solution concept returns the strategy $\hat{s}$ in $\mathcal{S}$ that obtains the highest average score from interaction with each member of $\mathcal{S}$. The essential difference between BSS and BITE is that BSS assumes $\mathcal{S}$ not to contain duplicates; thus, the choice of $\hat{s}$ does not depend upon the frequency with which different strategies might appear in the population.

5.1.2 Non-Monotonicity of BSS

Here we show that BSS is a non-monotonic solution concept. Our proof is by contradiction, so let us assume that BSS is monotonic. If BSS is monotonic, then by definition a complex $\mathcal{C}_x$ that is a solution to games A and C, where $A \supset C$, must also be a solution to any game B, where $A \supset B \supset C$. Let us consider a simple symmetric zero-sum game G_U. We can imagine the following sub-games of G_U:

$$C = W \cup \{x, y\} \tag{1}$$

$$B = C \cup V \tag{2}$$

$$G_U = A = B \cup U \tag{3}$$

where x and y are individual strategies, and U, V, W are sets (we treat A, B, C as games and sets). Equation 4 illustrates the resulting payoff matrix (bold numbers indicate sub-matrices of identical payoffs, e.g., $\mathbf{1}$ is all ones; '$*$' indicates that any sub-matrix can be used).

$$G_U = \begin{array}{c|ccccc} & x & y & W & V & U \\ \hline x & 0 & 0 & \mathbf{1} & \mathbf{-1} & \mathbf{1} \\ y & 0 & 0 & \mathbf{-1} & \mathbf{1} & \mathbf{-1} \\ W & \mathbf{-1} & \mathbf{1} & * & \mathbf{0} & \mathbf{0} \\ V & \mathbf{1} & \mathbf{-1} & \mathbf{0} & * & \mathbf{0} \\ U & \mathbf{-1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & * \end{array} \tag{4}$$

Let us say that x and y tie each other; further, x beats everyone in W and U, but loses to everyone in V, whereas y beats everyone in V, but loses to everyone in W and U. Also, let us say that members of W tie against members of V and U, and members of V tie against members of U (and W). The row-sum of scores for strategy x in game C is equal to $|W|$ (one win for each strategy in W). Similarly, the row-sum of scores for strategy y is $-|W|$. The row-sum of scores

in game C for any strategy in W must be less than the row-sum for strategy x: no strategy in W can accumulate more than $|W|-1$ wins (against members of W), since a strategy must tie itself. Thus, x is the solution to game C, and for x to be the solution to game A we need merely make $|U| + |W| > |V|$. Now, if BSS is a monotonic solution concept, then from the preceding it should immediately follow that x is also a solution to game B. But, if the cardinality of V is greater than that of W, then x cannot be the solution to game B; instead, we find that y is the solution. Thus, our assumption that BSS is monotonic is contradicted. □

The BSS solution concept is free of frequency-dependent effects, yet is still non-monotonic; thus, we can easily see that BITE—which is frequency-dependent—is also non-monotonic. The non-monotonicity of BSS and BITE is certainly consistent with the Red-Queen Effect [30] and with the common sentiment that objective metrics of goodness are difficult or impossible to obtain for coevolutionary systems (see, for example, [13]).

In an effort to grapple with this difficulty, research in [21] approaches the issue from the point of view of the game, discussing what properties a game should have to easily admit an objective measure of goodness; these properties restrict the game to a very narrow class that lacks the structures that give games of strategy much of their interest. In contrast, the work we present here shifts the focus of attention away from the game and onto the solution concept. Our preference relation (which provides a global partial order over the space of complexes) reveals that the difficulty in achieving monotonic performance can be due, in no small part, to the solution concept we choose.

5.2 Nash Equilibrium

5.2.1 Description

A Nash-equilibrium strategy in a symmetric game is a strategy that is its own *best reply*. Thus, given that both players have available to them the same set of strategies, if one player uses the Nash-strategy s^*, then the highest expected payoff obtainable by the other player is received when the other player also uses s^*. Stated more formally, s^* is a Nash-equilibrium strategy with respect to a strategy-set $\mathcal{S}$ if and only if $\forall s \in \mathcal{S} : G_{\mathcal{S}}(s, s^*) \leq G_{\mathcal{S}}(s^*, s^*)$.

A Nash equilibrium strategy may be either pure or mixed. For example, the Nash strategy for Rock-Paper-Scissors is a mixture where all three strategies are played with equal probability. Thus, the Nash equilibrium solution concept requires that behavior complexes allow collections of phenotypes (in contrast to BITE and BSS).

5.2.2 Monotonicity of Nash Equilibrium

Nash equilibrium is a monotonic solution concept in symmetric games. Our proof is by contradiction, so let us assume that the Nash-equilibrium concept is not monotonic. Figure 2 shows an example non-monotonic solution concept by which we prefer complex $\mathcal{C}_x$ to complex $\mathcal{C}_y$. From the definition of Nash equilibrium, we know that the strategy represented by $\mathcal{C}_x$ is its own best-reply in all of the games in $\mathcal{G}_x$. Since $\mathcal{C}_x$ is a solution for game E, we know that the strategies in $\mathcal{C}_x$ are found in E, as well as any superset of E. In particular, game D contains the strategies in $\mathcal{C}_x$. Figure 2 claims that $\mathcal{C}_x$ is not a solution to game D. Yet, because there exists no strategy in C that is a better reply to $\mathcal{C}_x$, there exists no better reply to $\mathcal{C}_x$ in any subset of C; particularly, a better reply to $\mathcal{C}_x$ cannot exist in D. Therefore, $\mathcal{C}_x$ must be a solution to game D, as well, in contradiction to the figure. Thus, Nash equilibrium must be a monotonic solution concept. □

Equation 5 shows a five-strategy, symmetric zero-sum game. Figure 3 shows the graph of the preference relation generated by this game using Nash equilibrium as the solution concept. Each "vertex" contains a behavior complex in bold and below it a list of games (or states of knowledge) for which the complex is a solution (each game is specified by the list of strategies in the game; each row is a separate game). A directed edge that connects two vertices indicates the preference relation. For example, we prefer the complex composed of strategies $\{bde\}$ to the complex $\{ce\}$—each game for which $\{ce\}$ is a solution is a sub-game of some game for which $\{bde\}$ is a solution. Since the preference relation is transitive, we also prefer $\{bde\}$ to $\{e\}$, for example. The reader can verify that no sequence of increasing states of knowledge will cause movement in the graph to go against the preference relation. The preference relation does allow movement between vertices that are not connected by an edge, but once a vertex is visited, neither it nor any of its descendents on the graph will be visited again (or at all).

$$G_U = \begin{array}{c|ccccc} & \text{a} & \text{b} & \text{c} & \text{d} & \text{e} \\ \hline \text{a} & 0 & 0 & 1 & 0 & -1 \\ \text{b} & 0 & 0 & -1 & -1 & 1 \\ \text{c} & -1 & 1 & 0 & -1 & 0 \\ \text{d} & 0 & 1 & 1 & 0 & -1 \\ \text{e} & 1 & -1 & 0 & 1 & 0 \end{array} \tag{5}$$

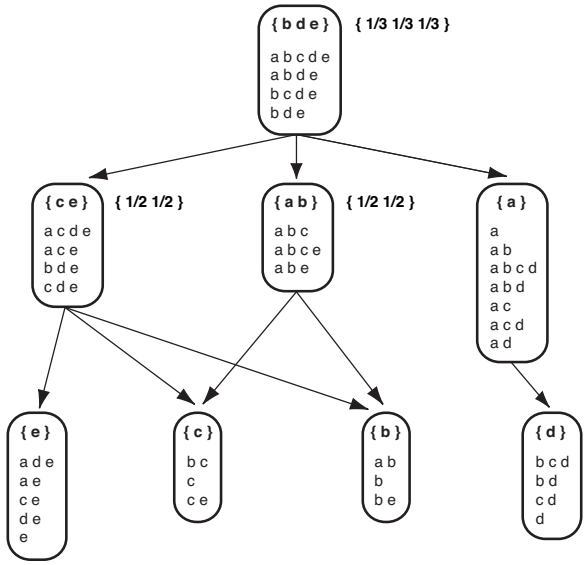

Figure 3: Preference diagram for game in Equation 5 using Nash equilibrium. Probability distributions for mixed-strategy $\mathcal{C}$ are indicated next to vertices.

6. DISCUSSION

The preference relation we define in Section 4 can be instantiated by any solution concept; limited space permits analysis of only two instantiations in this paper. Another

solution concept that is used in coevolutionary algorithms is *Pareto optimality*; we show in [9] that this concept is monotonic provided it is implemented in a specific way.

Throughout our analyses we assumed that strategies discovered by search are never thrown away. But, real-world algorithms have finite resources. Are our results relevant in this case? While infinite-memory assumptions simplify analysis, our earlier results on the *Nash memory mechanism* [10] show that monotonic improvement can be nicely approximated by a finite-memory system. Thus, initial data suggest that the benefits of monotonic concepts are robust.

The general notion of monotonic-improvement-over-time can, of course, be construed in different ways. An alternative view of monotonicity is given in studies of asymmetric zero-sum games between *learners* and *tests* [5, 6]. This work shows that a distance function to a global solution is monotonically decreased when a learner is discovered that solves a previously unsolved test (or combination of tests); that is, the minimal number of tests (or test combinations) that we *know* to be solvable increases monotonically. This alternative notion of monotonicity is subsumed by the one we present in this paper. Further, the guarantees offered by our notion of monotonicity extend beyond the known search-space into the unknown, as we discuss below.

7. OPEN-ENDED DOMAINS

Much research in coevolution concerns how a coevolutionary dynamic may generate an *open-ended* system, where novel and increasingly complex behaviors are continually innovated. Examples of such research include the evolution of language [15], the IPD [20, 16], competitive games in virtual worlds [28], artificial chemistries [12], and complex A-Life ecologies [25, 19, 29]. Nevertheless, a precise articulation of open-endedness is made difficult by, among other things, the lack of a universally agreed-upon notion of complexity [1, 3, 17]. From the point of view of the framework developed in this paper, we can contribute the following thoughts.

The game, as we formally define it here, is merely a function that maps pairs of strategies to payoffs; in particular, a game represents nothing about the *behavior* of a strategy, other than the payoffs it earns—a highly abstracted view of behavior, indeed. Thus, with respect to the theme of open-endedness, we can infer only certain things from a game. For example, if the game G_U is finite, then clearly the domain (or, more accurately, that part of the domain exposed by the evolving substrate) cannot be open-ended—there are only a finite number of (pure) strategies defined. Tic-Tac-Toe is an example of a closed-ended domain. Other domains, such as numbers games [31], are open-ended at least in the sense that they allow an infinity of strategies; for any finite set of numbers-game strategies, we can always find another strategy that will beat all of them, yet this winning strategy will appear no more complex than the others which it beats.

A more compelling form of open-endedness entails the generation of novel and unexpected solutions to the strategic challenges posed by coevolving individuals—open-endedness is a sequence of set-breaking tasks. Innovative solutions not only have intrinsic interest, but also reveal the domain to which they belong to have more nuance than previously imagined. We can thus make a simple taxonomy of domains: Closed-ended domains have a finite space of pure-strategy behaviors; *impoverished* open-ended domains have an infinity of behaviors, but these behaviors can be systematized in finite space—i.e., the "rules of the game" are finite; *rich* open-ended domains not only have an infinity of behaviors, but also the systematization of the domain requires infinite space. These categories of open-endedness echo ideas described in [29].

The idea of rich open-endedness appears to leave little room for objective metrics of goodness, since every new innovation changes the prism through which we view and understand the domain. For example, each innovation may uncover another intransitive structure in the domain; a strategy previously understood as "poor" may regain respectability because we realize that an entire aspect of its behavior was overlooked. But, we should not conflate behavioral complexity with adaptive utility. Whatever our perception of a strategy's behavior, the game of our formalism does not distinguish between impoverished and rich open-endedness, nor for that matter between closed-endedness and open-endedness.

Since our formalizations of solution concepts and monotonicity are built on the game, and not any deep representation of the domain's behaviors, our results concerning monotonicity apply equally to all domains, closed-ended and open-ended, impoverished and rich. If we do not discard information, then the solutions we obtain at time t are guaranteed to be no worse, in an objective (i.e., global) sense, than solutions obtained earlier; and, this is true *regardless of the fact that we have only local knowledge of the domain and no knowledge of what strategies we might discover in the future.* If our solution concept is non-monotonic, then no such guarantee can be made, even for closed-ended domains.

8. CONCLUSION

In this paper, we consider the question of whether a coevolutionary algorithm, when applied to solve a game of strategy, can monotonically improve the solution it reports as it operates over time, assuming that the algorithm is able to incrementally accumulate knowledge of the search space. Our intuitions do not provide a clear answer to this question. The monotonic increase of knowledge we assume suggests an affirmative answer, yet our familiarity with problems caused by intransitive superiority structures (such as that in Rock-Paper-Scissors) suggests a negative answer.

Our approach to this question is to first develop a preference relation over the search space. This preference relation creates a global partial-ordering. To answer the above question in the affirmative, we require that the solution concept implemented by the coevolutionary search algorithm never cause the algorithm's estimations to appear in a sequence that contradicts the partial order. A solution concept that guarantees such operation is monotonic.

We show that the solution concept conventionally implemented in a coevolutionary algorithm is not monotonic; that is, it does not allow us to expect monotonically improving estimations to be produced by the algorithm, *even with our assumption that knowledge is never discarded.* In contrast, we also show that the solution concept of Nash equilibrium is monotonic; if the algorithm implements Nash equilibrium, then it does guarantee monotonic improvement. Interestingly, this guarantee holds even though we assume nothing about the space of game strategies that remains to be discovered. Consequently, our results apply immediately to open-ended domains. Our formalism can be applied easily to other solution concepts, such as Pareto optimality [9].

Of course, our assumption that knowledge is never discarded is difficult to meet in the real-world. Nevertheless, our earlier work [10] demonstrates that a finite-memory system that heuristically discards knowledge can closely approximate monotonic improvement over time, if the solution concept allows.

9. ACKNOWLEDGMENTS

The author thanks Anthony Bucci, Edwin de Jong, Jordan Pollack, Shivakumar Viswanathan and members of the DEMO Lab.

10. REFERENCES

[1] C. Adami. What is complexity? *BioEssays*, 24(12):1085–1094, 2002.

[2] P. J. Angeline and J. B. Pollack. Competitive environments evolve better solutions for complex tasks. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 264–270. Morgan Kaufmann, 1993.

[3] J. P. Crutchfield. The calculi of emergence: Computation, dynamics and induction. *Physica D*, 75(1–3):11–54, 1994.

[4] P. Darwen and X. Yao. Automatic modularization by speciation. In T. Fukuda et al., editors, *Proc. of the 1996 IEEE International Conference on Evolutionary Computation*, pages 88–93. IEEE Press, 1996.

[5] E. D. de Jong. The incremental pareto-coevolution archive. In K. Deb et al., editors, *Proc. 2004 Genetic and Evolutionary Computation Conference*, LNCS 3102, pages 525–536. Springer, 2004.

[6] E. D. de Jong. The maxsolve algorithm for coevolution. In U.-M. O'Reilly, editor, *Proc. 2005 Genetic and Evolutionary Computation Conference*. ACM, 2005.

[7] K. A. De Jong. Are genetic algorithms function optimizers? In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 3–13. Elsevier, 1992.

[8] K. A. De Jong. Genetic algorithms are NOT function optimizers. In L. D. Whitely, editor, *Foundations of Genetic Algorithms 2 (FOGA)*, pages 5–18. Morgan Kaufmann, 1993.

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

[10] S. G. Ficici and J. B. Pollack. A game-theoretic memory mechanism for coevolution. In Cantú-Paz et al., editors, *2003 Genetic and Evolutionary Computation Conference*, pages 286–297. Springer, 2003.

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

[12] W. Fontana. Algorithmic chemistry. In Langton et al. [18], pages 159–209.

[13] H. Freund and R. Wolter. Evolution of bit strings: Some preliminary results. *Complex Systems*, 5:279–298, 1991.

[14] I. Harvey. Cognition is not computation: Evolution is not optimisation. In W. Gerstner et al., editors, *Proc. of the 7th Int. Conf. on Artificial Neural Networks*, pages 685–690. Springer-Verlag, 1997.

[15] T. Hashimoto and T. Ikegami. Emergence of net-grammar in communicating agents. *BioSystems*, 38(1):1–14, 1996.

[16] T. Ikegami. From genetic evolution to emergence of game strategies. *Physica D*, 75(1–3):310–327, 1994.

[17] J. F. Kolen and J. B. Pollack. The observer's paradox: Apparent computational complexity in physical systems. *Journal of Experimental and Theoretical Artificial Intelligence*, 7:253–277, 1995.

[18] C. Langton et al., editors. *Artificial Life II*. Addison-Wesley, 1992.

[19] R. E. Lenski, C. Ofria, T. C. Collier, and C. Adami. Genome complexity, robustness and genetic interactions in digital organisms. *Nature*, 400:661–664, August 1999.

[20] K. Lindgren. Evolutionary phenomena in simple dynamics. In Langton et al. [18], pages 295–312.

[21] S. Luke and R. P. Wiegand. When coevolutionary algorithms exhibit evolutionary dynamics. In A. Barry, editor, *2002 Genetic and Evolutionary Computation Conference Workshop Program*, pages 236–241, 2002.

[22] J. Maynard-Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982.

[23] S. Nolfi and D. Floreano. Co-evolving predator and prey robots: Do 'arm races' arise in artificial evolution? *Artificial Life*, 4(4):311–335, 1998.

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

[25] T. S. Ray. Evolution, complexity, entropy and artificial reality. *Physica D*, 75(1–3):239–263, 1994.

[26] C. D. Rosin and R. Belew. A competitive approach to game learning. In A. Blum and M. Kearns, editors, *Proc. of the Ninth Annual ACM Conf. on Computational Learning Theory*, pages 292–302. ACM Press, 1996.

[27] L. M. Schmitt. Theory of coevolutionary genetic algorithms. In M. Guo and L. T. Yang, editors, *Int. Symp. on Parallel and Distributed Processing and Applications*, volume 2745 of *LNCS*, pages 285–293. Springer, 2003.

[28] K. Sims. Evolving 3d morphology and behavior by competition. In R. A. Brooks and P. Maes, editors, *Artificial Life IV*, pages 28–39. MIT Press, 1994.

[29] T. Taylor. Creativity in evolution: Individuals, interactions and environments. In P. J. Bentley and D. W. Corne, editors, *Creative Evolutionary Systems*, pages 79–108. Morgan Kaufmann, 2001.

[30] L. van Valen. A new evolutionary law. *Evolutionary Theory*, 1:1–30, 1973.

[31] R. A. Watson and J. B. Pollack. Coevolutionary dynamics in a minimal substrate. In L. Spector et al., editors, *Proc. of the 2001 Genetic and Evolutionary Computation Conference*, pages 702–709, 2001.