Thoughts on Solution Concepts

Anthony Bucci
DEMO Lab
Michtom School of Computer Science
Brandeis University
Waltham, MA 02454
abucci@cs.brandeis.edu

Jordan B. Pollack
DEMO Lab
Michtom School of Computer Science
Brandeis University
Waltham, MA 02454
pollack@brandeis.edu

ABSTRACT

This paper explores connections between Ficici's notion of solution concept and order theory. Ficici postulates that algorithms should ascend an order called *weak preference*; thus, understanding this order is important to questions of designing algorithms. We observe that the weak preference order is closely related to the pullback of the so-called lower ordering on subsets of an ordered set. The latter can, in turn, be represented as the pullback of the subset ordering of a certain powerset. Taken together, these two observations represent the weak preference ordering in a more simple and concrete form as a subset ordering. We utilize this representation to show that algorithms which ascend the weak preference ordering are vulnerable to a kind of bloating problem. Since this kind of bloat has been observed several times in practice, we hypothesize that ascending weak preference may be the cause. Finally, we show that monotonic solution concepts are convex in the order-theoretic sense. We conclude by speculating that monotonic solution concepts might be derivable from non-monotonic ones by taking convex hull. Since several intuitive solution concepts like average fitness are not monotonic, there is practical value in creating monotonic solution concepts from non-monotonic ones.

Categories and Subject Descriptors

G.1.6 [**Mathematics of Computing**]: Optimization; I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic Methods

General Terms

Algorithms,Theory

Keywords

bloat, coevolution, coevolutionary algorithms, later is better, solution concepts, theory

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

1. INTRODUCTION

The historical arc which coevolutionary algorithms research has taken since the early 1990s can be said to have passed from practice to theory. Well-cited empirical results such as Daniel Hillis' coevolution of sorting networks [8] or Karl Sims' coevolution of battling virtual creatures [14] inspired early work. The growing list of successful applications of cooperative coevolutionary algorithms [11] stoked further enthusiasm. However, equally well-known undesirable algorithm behavior, ranging from cycling to overspecialization to relative overgeneralization[1] eventually caused enough confusion that research attention shifted to developing theory.

The 2002 Genetic and Evolutionary Computation Conference hosted a workshop entitled *Understanding Coevolution: Theory and Analysis of Coevolutionary Algorithms* which marked a shift from practice to theory. The description of this workshop put it simply that "dynamics in these systems can be complicated and surprising" but did go on to say that "the time has come to focus our collective attention on analysis issues."

Sevan Ficici's 2004 Ph.D. dissertation grounds the analysis of coevolutionary dynamics in two key ideas, the *solution concept* and the *weak preference* order. Regarding solution concepts, Ficici says:

> We assert that pathologies in coevolutionary optimization arise when algorithms fail to implement the required (or desired) solution concepts. [5]

The implication is that an algorithm which does implement a solution concept will behave. Yet there is more to the story. [5] also introduces the notion of a *non-monotonic* solution concept and argues that an algorithm which faithfully implements one can still exhibit pathological behavior, namely cycling. Roughly speaking, an algorithm implementing a non-monotonic solution concept can gain or lose capabilities much as a non-monotonic function can fluctuate up or down. Ficici shows that intuitive and commonly-used solution concepts such as "highest average fitness" are non-monotonic when used in coevolutionary algorithms. Hence, algorithms implementing non-monotonic solutions concepts are *predicted* to produce poor algorithm dynamics, and in particular to be vulnerable to cycling.

Weak preference is an ordering among candidate solutions which can be derived from any solution concept, but is most meaningful when derived from a monotonic solution concept. In that case, the weak preference order is such that

[1]See [17] for a discussion of relative overgeneralization.

ascending it yields a chain of candidates with increasing capability. Thus, theoretically speaking we might ground the aim of "implementing a (monotonic) solution concept" in terms of whether the algorithm prefers individuals which are higher on the weak preference order.

This paper explores these ideas from a formal standpoint, continuing along the theoretical arc of research but with an eye towards improving and understanding algorithms. While we do not advance a particular claim, we do make two loosely-connected observations which have implications for algorithm design:

1. That under certain conditions, even a supposedly well-behaved algorithm which ascends the weak preference order and implements a *monotonic* solution concept can still exhibit a kind of pathological bloating phenomenon. We have named this *the later is better effect* and suggested examples where it may have arisen.
2. That monotonicity of solution concepts is a kind of convexity. We speculate that the formal convex hull operation will yield a way to convert a non-monotonic solution concept into a monotonic one.

We make these observations by giving solution concepts and weak preference a concrete grounding in the theory of ordered sets.

This paper is organized as follows. Section 2 develops the mathematical notation and concepts used in the remainder of the paper. Section 3 gives background on the notion of solution concept. Section 4 defines the weak preference relation, makes the connection between weak preference and the lower ordering on subsets, and argues for the possibility of *the later is better effect*. Finally, section 5 speculates on the possibility of converting a non-monotonic solution concept into a better-behaved monotonic one via the order-theoretic equivalent of the convex hull operation.

2. MATHEMATICAL PRELIMINARIES

This section reviews some concepts from the theory of ordered sets. For a gentle introduction to ordered sets, see [13]. Readable treatments of more advanced concepts like pullbacks can be found in [1] or [15]. The latter book covers the lower/upper orders and convexity in some detail.

Definition 2.1. (**Ordered Set**) A set S equipped with a binary relation $\leq_S$ is called *ordered* when $\leq_S$ is both reflexive and transitive. That is, for all $s \in S$, $s \leq_S s$; and, for all $s_1, s_2, s_3 \in S$, if $s_1 \leq_S s_2$ and $s_2 \leq_S s_3$, then $s_1 \leq_S s_3$. When S is an ordered set we will refer to it simply by S; when confusion might arise, we will write $(S, \leq_S)$.

Remark 2.2. An order on S is *symmetric* if $a \leq_S b$ and $b \leq_S a$ together imply $a = b$ for any a and b in S. What we are calling an order need not be symmetric, however. Thus, we will define a strict version of the order like so: $a <_S b$ if $a \leq_S b$ and $b \not\leq_S$.

Definition 2.3. (**Pullback Order**) Let S be an ordered set, A some other set, and $f : A \to S$ an arbitrary function into S. Then we can make A into an ordered set by *pullback*[2], as follows: define $a_1 \leq_f a_2$ to hold in A whenever $f(a_1) \leq_S f(a_2)$ in S.

[2]So-called to suggest the action of *pulling* the order of S *back* through the function f. But the technical notion of pullback applies here too; see, for instance, [1].

LEMMA 2.4. *Let S be an ordered set, $f : A \to S$ a function into S. Then $(A, \leq_f)$ really is an ordered set.*

PROOF. $\leq_S$ is reflexive, so that $f(a) \leq_S f(a)$ for any $a \in A$; thus, $a \leq_f a$, meaning $\leq_f$ is reflexive too. For transitivity, let $a_1, a_2, a_3 \in A$ be such that $a_1 \leq_f a_2$ and $a_2 \leq_f a_3$. That is, $f(a_1) \leq_S f(a_2)$ and $f(a_2) \leq_S f(a_3)$. By transitivity of $\leq_S$, $f(a_1) \leq_S f(a_3)$, meaning $a_1 \leq_f a_3$. Thus $a_1 \leq_f a_3$, showing $\leq_f$ is transitive too. Thus $(A, \leq_f)$ is indeed an ordered set. □

Remark 2.5. One reason to labor over the proof that pullback preserves reflexivity and transitivity is that it does not preserve symmetry: if $\leq_S$ is symmetric, $\leq_f$ will not be whenever f is not injective.

Definition 2.6. (**Lower Ordering**) Let S be an ordered set, $\mathscr{P}(S)$ the power set of S. The *lower ordering* of $\mathscr{P}(S)$, written $\leq^\flat$, is defined as follows: for any $S_1, S_2 \subset S$, $S_1 \leq^\flat S_2$ holds if, for all $s_1 \in S_1$, there is some $s_2 \in S_2$ such that $s_1 \leq_S s_2$.

Definition 2.7. (**Downward Closure**) Let S be an ordered set, $U \subset S$ some subset of S. The *downward closure* of U, written $\downarrow U$, is the set $\{s \in S \mid s \leq_S u \text{ for some } u \in U\}$.

Remark 2.8. There is a dually-defined *upward closure* of U, written $\uparrow U$.

LEMMA 2.9. *Let S be an ordered set, $U, V \subset S$. Then $U \leq^\flat V$ if and only if $\downarrow U \subset \downarrow V$.*

PROOF. ($\Rightarrow$): Assume $U \leq^\flat V$. Let $x \in \downarrow U$; we must show $x \in \downarrow V$ also to prove $\downarrow U \subset \downarrow V$. $x \in \downarrow U$ means there is a $u \in U$ such that $x \leq_S u$. Since $U \leq^\flat V$, there is some $v \in V$ such that $u \leq_S v$. $\leq_S$ is transitive, so $x \leq v$. Thus, $x \in \downarrow V$. This fact holds for any $x \in \downarrow U$, implying $\downarrow U \subset \downarrow V$.

($\Leftarrow$): Conversely, assume $\downarrow U \subset \downarrow V$ and let $u \in U$. By assumption, $u \in \downarrow V$. By the definition of $\downarrow V$, there is some $v \in V$ such that $u \leq_S v$. This holds for any $u \in U$, which means we have shown that for any $u \in U$, there is some $v \in V$ with $u \leq_S v$, or equivalently that $U \leq^\flat V$. □

LEMMA 2.10. *Let S be an ordered set. Then $(\mathscr{P}(S), \leq^\flat)$ is an ordered set which differs, in general, from $(\mathscr{P}(S), \subset)$.*

PROOF. Note that the mapping $U \mapsto \downarrow U$ is a function $\downarrow(-) : \mathscr{P}(S) \to \mathscr{P}(S)$. Lemma 2.9 says exactly that $\leq^\flat$ is the pullback of $\subset$ through $\downarrow(-)$. Hence, by Lemma 2.4, $\leq^\flat$ really is an order on $\mathscr{P}(S)$. If $a \neq b$ are two distinct elements of S such that $a \leq b$, then $\{a\} \not\subset \{b\}$, but $\{a\} \leq^\flat \{b\}$, meaning in general $\subset \neq \leq^\flat$ as orders on $\mathscr{P}(S)$. □

Definition 2.11. (**Convex Subset**) Let S be an ordered set, $U \subset S$ some subset. U is *convex* if, whenever $s_1 \leq_S s \leq_S s_2$ with $s_1, s_2 \in R$, then $s \in U$ also. That is, all elements of S which lie between elements of U are also in U.

Example 2.12. The set of even numbers is not a convex subset of $(\mathbb{N}, \leq)$ because, for instance, $2 \leq 3 \leq 4$ but 3 is not even. On the other hand, an interval like $[2, 10]$ is convex.

Definition 2.13. (**Convex Hull**) Let S be an ordered set, $U \subset S$. The *convex hull* of U in S, written $\widehat{U}$, is defined $\widehat{U} = \{s \in S \mid \text{there are } u_1, u_2 \in U \text{ such that } u_1 \leq_S s \leq_S u_2\}$.

Example 2.14. The convex hull of the set of even numbers in $\mathbb{N}$ is $\mathbb{N}$ itself. The convex hull of the interval $[2, 10]$ is $[2, 10]$ itself. Likewise, the convex hull of $\{2, 4, 6, 8, 10\}$ is also $[2, 10]$.

LEMMA 2.15. *Let S be an ordered set, $U \subset S$. Then* $\widehat{U} = {\downarrow}U \cap {\uparrow}U$.

PROOF. ($\subset$): Let $u \in \widehat{U}$. Then there are $u_1, u_2 \in U$ such that $u_1 \leq_S u \leq_S u_2$. Thus $u \in {\uparrow}U$ (as a result of $u_1 \leq_S u$), and $u \in {\downarrow}U$ (as a result of $u \leq_S u_2$). In other words, $u \in {\downarrow}U \cap {\uparrow}U$. This fact holds for all $u \in \widehat{U}$, showing $\widehat{U} \subset {\downarrow}U \cap {\uparrow}U$.

($\supset$): Let $u \in {\downarrow}U \cap {\uparrow}U$. $u \in {\downarrow}U$ implies there is a $u_2 \in U$ such that $u \leq_S u_2$. Similarly, $u \in {\uparrow}U$ implies there is a $u_1 \in U$ such that $u_1 \leq_S u$. Thus, $u_1 \leq_S u \leq_S u_2$, meaning $u \in \widehat{U}$. This holds for all $u \in {\downarrow}U \cap {\uparrow}U$, so ${\downarrow}U \cap {\uparrow}U \subset \widehat{U}$. □

3. SOLUTION CONCEPTS

As originally defined in [5], a *solution concept* is a binary predicate on the collection of candidate solutions buildable from any population. It specifies which of the available candidate solutions is to be regarded as an actual solution at that point in evolutionary time.

As a simple example from evolutionary algorithms, consider an objective function $f : G \to \mathbb{R}$ from some set of genotypes G to the real numbers. If $G_t \subset G$ is considered to be the population at time t we might take the set

$$\partial G_t = \arg\max_{g \in G_t} f(g) \tag{1}$$

as the subset of G_t which we regard as (provisional) solutions at this point in evolutionary time. The boundary operator ∂X denotes "set of solutions in context X." Equation 1 is the mathematical expression of a common solution concept, the "best so far" or "maximum fitness" concept. Notice that an optimization problem is ill-defined without some solution concept; stating "solve $f : G \to \mathbb{R}$" has no meaning until we know we are to find the maxima, and more specifically know that we seek, in this case, all of ∂G. The fundamental hope behind local search in general and evolutionary algorithms in particular is that knowledge of local solutions ∂G_t can direct an algorithm towards the global solutions in ∂G.

Ficici's notion of solution concept is a generalization of this idea to coevolutionary algorithms. The interactive nature of the problem domains explored by coevolutionary algorithms adds a level of complication to the definition of solution which is not often encountered in more conventional optimization problems. In this section we will review solution concepts, simultaneously developing a notation which we feel simplifies and clarifies that used in [5].

Throughout this section, we assume that the interactive domain under scrutiny is expressed by two functions $p_S : S \times T \to R_S$ and $p_T : S \times T \to R_T$. We interpret the sets S and T to be two different *roles* which entities can play. R_S and R_T are two ordered sets. $p_S(s, t)$ is interpreted as the value from R_S which is assigned to the element playing role S, namely s[3], after it interacts with t.[4] Similar statements apply to p_T.[5] Sometimes it will be convenient to ignore the distinction between S and T. Note, for instance, that when we discuss populations we can consider collections formed from $S + T$, the disjoint union of S and T. For the sake of notational brevity we will therefore sometimes write X instead of $S + T$.

3.1 Entities and Configurations

The sets S and T contain what might be thought of as atomic entities. Populations are collections, be they sets, multisets, or distributions, drawn from $S+T$. Consequently, the elements of S and T are the *units of search*. It often happens that the domain does not have its best solutions among the S or T, though, so that algorithms form *configurations* of these as candidate solutions. We will write $C(X)$ for the set of configurations built from a set X. $C(X)$ can be interpreted as a kind of free structure, containing all possible combinations of objects drawn from X. For example:

Cooperative Coevolution.

Cooperative coevolutionary algorithms [12] might maintain two populations, one drawn from S and one drawn from T. The aim is usually to find a pair (s, t) which maximizes a given objective function $f : S \times T \to \mathbb{R}$. Here, then, $C(X) = C(S + T) = S \times T$.

Nash Memory.

[5], for example, discusses algorithms which attempt to find mixtures of entities, one mixture for each role. If we use the simplex notation Λ^S to represent the set of all mixtures over S (as in [16]), then $C(X) = C(S + T) = \Lambda^S \times \Lambda^T$.

Pareto Coevolution.

Pareto coevolutionary algorithms [6, 10] seek the non-dominated front of a set of entities S taken over a set of objectives T. Given a function like $p : S \times T \to R$, R being ordered, we can treat each $t \in T$ as if it were a function $S \to R$ by currying: $t(s) = p(s, t)$ (see [2] for details on this point of view). Thus, our solutions are $C(X) = C(S+T) = \mathscr{P}(S)$. If we also seek a subset of T, such as the maximally-informative tests [2] or a complete evaluation set [4], then we might instead use $C(X) = \mathscr{P}(S) \times \mathscr{P}(T)$.

3.2 Solutions

A *solution concept* designates a subset of the set of configurations as solutions. The concept should specify a subset for all possible sets of configurations. If $S_t \subset S$ and $T_t \subset T$ are the population(s)[6] at time t, then $C(S_t + T_t)$ is the set of configurations at time t; a solution concept must specify a subset for all possible S_t and T_t.[7]

We will write ∂X_t to designate the solution set of X_t.

[3]Which Ficici would call a *behavior* chosen for that role.
[4]In the *event* (s, t).
[5]One or more of these objects may be identical; for instance, we might have $S = T$. We wish to avoid results which are contingent on assumptions like that, however, which is why we adopt this general formulation.
[6]To be more precise, we should say that S_t and T_t are the *supports* of the populations, as populations can be multisets.
[7]Observe that the concept need not specify solutions for arbitrary subsets of $C(S + T)$, only for those expressible as $C(S_t + T_t)$. The distinction is acute in the case of simplices, where if $X_t \subset X$, then $C(X_t) \subset C(X)$ can only be an $|X_t|$-dimensional *face* of the $|X|$-dimensional simplex of configurations. Most subsets of $C(X)$ are not of this form.

We omit the $C(-)$ for brevity, so that $\partial X_t \subset C(X_t)$ is the set of provisional solutions at time t.[8] Thus, ∂ itself is the solution concept. We use the boundary operator ∂ to suggest that a solution concept is giving a frontier or edge to the population.

4. PREFERENCE AND BLOAT

A population can be viewed as a context within which configurations are compared. However, the context is local in the sense that only the configurations which appear together (that is, are buildable from that population's support) can be compared. The *weak preference* relation is meant to convert these local comparisons into a global one. That is, weak preference allows one to compare two configurations which may not appear in the same contexts together.

Simply put, a configuration β is preferred to α if, whenever α appears to be a solution in one context, there is a larger context in which β appears as a solution also. In this section we will give the formal definition of weak preference and show that it is closely related to the lower ordering on the powerset of the set of all possible populations. This observation will allow us to represent weak preference as the subset ordering on that powerset, via lemma 2.9 and pullback through the mapping of configurations to the set of populations in which they appear optimal.

Next we observe that weak preference is vulnerable to a kind of bloating problem. We show that it is possible to weakly prefer β to α even when β never shows up as a solution where α does not. This result highlights that weak preference tends to prefer configurations which appear in larger populations. Larger populations, which have larger support sets, give rise to configurations over larger support sets[9], which in turn means that configurations which appear in these larger populations but not in smaller ones will have larger support. They will be preferred by weak preference, meaning an algorithm which ascends weak preference will replace smaller-support configurations with larger-support configurations even when the latter do not perform better. Finally, we give three examples where a kind of bloat has occurred and suggest these may all be consequences of the bloat phenomenon predicted by this theory.

4.1 Weak Preference

Section 2.6.4 of [5] defines weak preference as follows:

Definition 4.1. (**Weak Preference**) Let α and β be configurations, and let $X = S + T$ be the set of all possible entities. β is *weakly preferred* to α, which we will write $\alpha \prec \beta$ if, for any context $X_\alpha \subset X$ such that $\alpha \in \partial X_\alpha$, there is a strictly larger X_β (meaning $X_\alpha \subset X_\beta$ and $X_\alpha \not\subset X_\beta$) such that $\beta \in \partial X_\beta$.

Remark 4.2. Note that there is no condition saying that when $\beta \in \partial X_\beta$ that $\alpha \notin \partial X_\beta$. In other words, it is possible in principle that in every population X_β in definition 4.1, α also appears.

For any configuration α, let U_α be the set of all populations in which α appears as a solution. Symbolically,

$$U_\alpha = \{X' \subset X \mid \alpha \in \partial X'\} \tag{2}$$

For any α, U_α consists of those contexts ($X' \subset X$) in which α appears as a solution, so that U_α is a set of subsets of X (the set of all possible entities). In symbols, $U_\alpha \subset \mathscr{P}(X)$ or, equivalently, $U_\alpha \in \mathscr{P}(\mathscr{P}(X))$. $\mathscr{P}(X)$ is ordered by inclusion so that, by definition 2.6, we can order $\mathscr{P}(\mathscr{P}(X))$ by the lower ordering $\preceq^\flat$.

In particular, we can compare any two U_α and U_β via the lower order on $\mathscr{P}(\mathscr{P}(X))$. $U_\alpha \preceq^\flat U_\beta$ if, for any context $X_\alpha \in U_\alpha$, there is a context $X_\beta \in U_\beta$ such that $X_\alpha \subset X_\beta$. This observation is essentially the content of:

Definition 4.3. (**Configuration Order**) Let α and β be two configurations. Define a relation on configurations, which we will write $\preceq^\flat$, as follows: $\alpha \preceq^\flat \beta$ when $U_\alpha \preceq^\flat U_\beta$ as elements of $\mathscr{P}(\mathscr{P}(X))$.

POSTULATE 4.4. *As a relation on configurations, $\preceq^\flat$ really is an order.*

PROOF. $\preceq^\flat$ is the pullback of the lower order on $\mathscr{P}(\mathscr{P}(X))$ through the mapping $\alpha \mapsto U_\alpha$. The lower order is an order, by lemma 2.10, and the pullback of an order is also an order by lemma 2.4. Thus $\preceq^\flat$ is an order. □

We have purposely used the symbol $\preceq^\flat$ to suggest the weak preference order $\prec$, because:

THEOREM 4.5. *Assume the problem domain is such that for all configurations α, U_α is bounded in the sense that for any $X_\alpha \in U_\alpha$ and any chain of inclusions $X_\alpha = X_0 \subset X_1 \subset X_2 \subset \ldots$, there is an N such that for all $k \geq N$, $X_k = X_{k+1}$. Assume also that whenever a configuration α is in both ∂X_1 and ∂X_2, it is also in $\partial(X_1 \cup X_2)$. Then $\alpha \prec \beta$ if and only if $\alpha \prec^\flat \beta$.*

PROOF. ($\Rightarrow$): $\alpha \prec \beta$ immediately implies $\alpha \preceq^\flat \beta$. According to remark 2.2, we must show it also implies $\beta \not\preceq^\flat \alpha$. Imagine that $X_\alpha \in U_\alpha$ and $X_\beta \in U_\beta$ are such that $X_\alpha \subset X_\beta$ strictly (meaning $X_\alpha \neq X_\beta$), as required by the assumption that $\alpha \prec \beta$. If it were the case that $\beta \preceq^\flat \alpha$, there would in turn be a $X'_\alpha \in U_\alpha$ such that $X_\beta \subset X^1_\alpha$. $X_\alpha \subset X_\beta$ strictly, implying $X_\alpha \subset X^1_\alpha$ strictly also. By repeating this argument, we can produce an infinite, strictly-increasing chain of subsets $X_\alpha \subset X^1_\alpha \subset X^2_\alpha \subset \ldots$, contradicting the assumption that the domain has no such chains. By this contradiction, we cannot have $\beta \preceq^\flat \alpha$, and we have shown that $\alpha \prec \beta$ implies $\alpha \prec^\flat \beta$.

($\Leftarrow$): Imagine $\alpha \prec^\flat \beta$, meaning $\alpha \preceq^\flat \beta$ and $\beta \not\preceq^\flat \alpha$. We wish to show $\alpha \prec \beta$. Firstly, $\beta \not\preceq^\flat \alpha$ implies that there is at least one $X^*_\alpha \in U_\alpha$ such that there is a $X^*_\beta \in U_\beta$ with $X^*_\alpha \subset X^*_\beta$ and such that there is no other $\bar{X}_\alpha \in U_\alpha$ with $X^*_\beta \subset \bar{X}_\alpha$. Now imagine $X_\alpha \in U_\alpha$. Since $\alpha \preceq^\flat \beta$, there is a $X_\beta \in U_\beta$ with $X_\alpha \subset X_\beta$. Either this relationship is strict, or $X_\alpha = X_\beta$. But in the latter case, it must be that $X_\alpha \subset X_\beta \cup X^*_\beta$ strictly; otherwise, we would have $X_\alpha = X^*_\beta$, contradicting the assumption that there is no such X_α. By assumption, $X_\beta \cup X^*_\beta \in U_\beta$, which means that for any $X_\alpha \in U_\alpha$ there is always a strictly larger context in U_β. Thus we have that $\alpha \prec \beta$, as we set out to show. □

Remark 4.6. In particular, the theorem holds when the global set of entities X is finite, because any chain of subsets

[8] Recall that this is written $\mathrm{S}^*(\mathbf{T}, \mathcal{O})$ in [5], where $\mathbf{T}$, a measurement table, corresponds to our S_t, T_t and the associated subfunctions of p_{S} and p_{T}, $\mathcal{O}$ represents the optimality concept, and S^* simply means to apply the concept to the table.

[9] For instance, when configurations are mixtures.

$X_i \subset X_{i+1} \subset \ldots$ will, at worst, top out at X. The condition that if β is a solution in two populations, then it is also a solution in their union, appears to hold in common solution concepts. This theorem says that in such instances, the strict version of configuration ordering (definition 4.3; see remark 2.2 for discussion of strictness) corresponds exactly to the weak preference ordering (definition 4.1). In short, for all practical purposes these two are the same order.

4.2 The Potential for Bloat

Remark 4.2 suggested that we could have $\alpha \prec \beta$, β weakly preferred to α, even if it is never the case that when $X_\alpha \in U_\alpha$ and $X_\beta \in U_\beta$ are such that $X_\alpha \subset X_\beta$, that $\alpha \notin \partial X_\beta$. In fact, it may be that all such ways of enlarging a context in which α is a solution to produce a context in which β is a solution always leave α as a solution as well. Nevertheless, we would prefer β to α in an instance like this. We could sum up this effect with the maxim "later is better" according to weak preference.

While it remains to be seen whether there are non-trivial examples of this *later is better effect*, we feel it is worth looking at more closely because:

- In common cases, we would expect later is better to produce configurations which are bloated, in a sense;
- This kind of bloat has been observed at least once in practice.

Later Is Better Can Lead to Bloat.

Imagine $\alpha \prec \beta$ is an instance of the later is better effect. Firstly, note that β appears as a solution strictly after α. In particular, there are populations in which α appears as a solution but β does not. One example where this may occur, in the case of mixture configurations, is when β is a mixture with the same support as α but including more atomic entities. β would appear later because the extra members of its support would have to be discovered. If, then α and β both appear as solutions, β would be preferred. While this may be precisely what we want in some cases[10], it may be exactly what we do not want in other cases. For instance, if an algorithm is continually generating new entities which are different from existing ones but do not improve performance, mixing solution configurations α with these can produce configurations β which have strictly larger support but are no better in any other sense.

Possible Observations of Bloat.

- Anecdotally, coevolution of Tic Tac Toe strategies seems to show this effect. Algorithms have a tendency to produce strategies which can draw existing strategies, but not win. Thus, over time, large numbers of drawing strategies are found and collected into configurations which grow in size but do not improve performance-wise.[11]
- In evolutionary approaches to multi-objective optimization [7], one finds that as the number of objectives grows, finding candidate solutions which dominate existing solutions becomes increasingly difficult. In Pareto coevolution, therefore, one would expect that if an algorithm begins accumulating different tests[12], the non-dominated front would begin to grow out of control. However, the front is not necessarily gaining any performance advantanges in this case, and therefore might be bloating.
- The LAPCA algorithm has been proposed as a solution concept for Pareto coevolution [3]. Recent work with this algorithm coevolving players for a modified version of Pong [9] suggests a bloating phenomenon. While that work did not directly test for bloat the results in Table 3 reporting the number of evaluations used seem to suggest an increase in the size of the layered archive beyond what might be reasonably expected.[13] The extra evaluations suggest more individuals than necessary were stored in the layered archive. This same work argues more definitely that a bloating phenomenon occurs with best of generation methods.

It is worth emphasizing that the later is better effect may plague even monotonic solution concepts. Definition 5.1 gives the formal definition of these, but note:

1. None of the preceding relied on the solution concept being non-monotonic.
2. Pareto coevolution implements a monotonic solution concept [5], and LAPCA has been argued to approximate one as well [3].

5. DISCUSSION

Thus far we have detailed a representation of Ficici's weak preference order as a subset relation in a certain powerset. The concreteness of the subset relation led to several observations about weak preference which might not have been easily elucidated otherwise. We feel that with further work, deeper predictive applications of this representation might be developed. Here we hint at one such possibility regarding the convex hull of a solution concept. First we observe that a monotonic solution concept has a certain convexity property. Next, we observe that any solution concept at all will involve sets which have a convex hull. Finally, we speculate that it might be possible to close even an ill-behaved, non-monotonic solution concept into a corresponding monotonic one which, according to [5], would be expected to have better dynamic properties. Such a result would have practical significance, as algorithm designers would not need to worry over theoretical details like monotonicity.

To begin, note that as defined in section 9.2.2 of [5], a *monotonic solution concept* is as follows:

Definition 5.1. (**Monotonic Solution Concept**) Let α be a configuration. Whenever $\alpha \in \partial X_1$, $\alpha \notin \partial X_2$ and $\alpha \in \partial X_3$, where $X_1 \subset X_2 \subset X_3$, ∂ is said to be *non-monotonic*. Otherwise, ∂ is *monotonic*.

In words, this definition says that whenever α appears as a solution in two contexts where one includes the other, then

[10] For instance, in rock-scissors-paper, the atomic strategy "always play rock" would appear before the mixture "play rock, scissors or paper each with probability $\frac{1}{3}$."

[11] Sevan Ficici, personal communication.

[12] *Tests* as in coevolving individuals which are treated as objectives; see, for instance, [6, 10].

[13] LAPCA uses significantly more evaluations than a BOG (best of generation) method.

α must also appear as a solution in all contexts included between them. In short, ∂ is *convex* in the following sense:

THEOREM 5.2. *Let ∂ be a solution concept. Then ∂ is monotonic if and only if U_α is convex in $(\mathscr{P}(\mathscr{P}(X)), \subset)$ for all configurations α.*

PROOF. ∂ is monotonic if and only if for all α, whenever $X_1, X_3 \in U_\alpha$ and $X_1 \subset X_3$, then any X_2 with $X_1 \subset X_2 \subset X_3$ must also be in U_α. This equivalence is really a formal restatement of definition 5.1. But this condition is also exactly that for all α, the set U_α is convex, in the sense of definition 2.11, with respect to $\subset$. U_α lives in $\mathscr{P}(\mathscr{P}(X))$; thus, ∂ is monotonic if and only if U_α is convex for all configurations α. □

Now imagine that ∂ is not monotonic. Then for at least one configuration α, U_α is not a convex set.[14] However, the convex hull construction defined in definition 2.13 can be applied to yield $\widehat{U_\alpha}$ for this α.

Naturally this is only a theoretical construction, but in our opinion the question of whether it can be made practical is worth exploring. The change from U_α to $\widehat{U_\alpha}$ explicitly adds to U_α all the contexts in which α "should be" a solution, but according to ∂ is not. There may be intractably many contexts with that quality, and it may be infeasible to discover what those are. However, by studying small domains we hope to elucidate principles which allow us to systematically produce the convex hull of any given solution concept. It may be that a known change, for instance from average fitness to Pareto dominance, already implements this convex hull operation. Would an algorithm which implemented this new monotonic solution concept be solving the original problem, or would it be exploring an essentially different domain?

Regarding bloat, is there any hope of avoiding the later is better effect? An obvious ploy is to prefer parsimonious solutions. However, doing so changes the preference order among candidates and thus implicitly changes the solution concept implemented by the algorithm. What other implications does such a change have?

Further study is necessary to clarify these questions. In future work we hope to provide principled and theoretically-grounded answers to the questions of how to avoid bloat and how to take the convex hull of a given solution concept.

6. REFERENCES

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

[2] A. Bucci and J. B. Pollack. A Mathematical Framework for the Study of Coevolution. In K. De Jong, R. Poli, and J. Rowe, editors, *FOGA 7: Proceedings of the Foundations of Genetic Algorithms Workshop*, pages 221–235, San Francisco, CA, 2003. Morgan Kaufmann Publishers.

[3] E. D. De Jong. Towards a Bounded Pareto-Coevolution Archive. In *Proceedings of the Congress on Evolutionary Computation – CEC'2004*, volume 2, pages 2341–2348. IEEE Service Center, 2004.

[4] E. D. De Jong and J. B. Pollack. Ideal Evaluation from Coevolution. *Evolutionary Computation*, 12(2), 2004.

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

[6] S. G. Ficici and J. B. Pollack. Pareto Optimality in Coevolutionary Learning. In *European Conference on Artificial Life*, pages 316–325, 2001.

[7] C. M. Fonseca and P. J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[8] D. W. Hillis. Co-evolving Parasites Improve Simulated Evolution in an Optimization Procedure. *Physica D*, 42:228–234, 1990.

[9] G. A. Monroy, K. O. Stanley, and R. Miikkulainen. Coevolution of Neural Networks Using a Layered Pareto Archive. In *GECCO '06: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, pages 329–336. ACM Press, 2006.

[10] J. Noble and R. A. Watson. Pareto Coevolution: Using Performance Against Coevolved Opponents in a Game as Dimensions for Pareto Selection. In L. Spector et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, pages 493–500, San Francisco, CA, 2001. Morgan Kaufmann Publishers.

[11] M. A. Potter and K. A. De Jong. A Cooperative Coevolutionary Approach to Function Optimization. In *Parallel Problem Solving from Nature*, pages 249–257, 1994.

[12] M. A. Potter and K. A. De Jong. Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.

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

[14] K. Sims. Evolving 3D Morphology and Behavior by Competition. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 28–39, Cambridge, MA, 1994. The MIT Press.

[15] P. Taylor. *Practical Foundations of Mathematics*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, UK, 1st edition, 1999.

[16] M. D. Vose. *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge, MA, 1999.

[17] R. P. Wiegand. *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, George Mason University, Fairfax, Virginia, 2003.

[14] Note that we can define a *local* monotonicity at α to mean U_α is convex. In light of that, the monotonicity criterion may be far too strict. An algorithm implementing a solution concept which is locally monotonic for *enough* configurations may not exhibit bad behavior.