# A No-Free-Lunch Framework for Coevolution

Travis C. Service
Missouri University of Science and Technology
325 Computer Science Building
500 West 15th Street
Rolla, Missouri 65409-0350, USA
tservice@acm.org

Daniel R. Tauritz
Missouri University of Science and Technology
324 Computer Science Building
500 West 15th Street
Rolla, Missouri 65409-0350, USA
dtauritz@acm.org

## ABSTRACT

The No-Free-Lunch theorem is a fundamental result in the field of black-box function optimization. Recent work has shown that coevolution can exhibit free lunches. The question as to which classes of coevolution exhibit free lunches is still open. In this paper we present a novel framework for analyzing No-Free-Lunch like results for classes of coevolutionary algorithms. Our framework has the advantage of analyzing No-Free-Lunch like inquiries in terms of solution concepts and isomorphisms on the weak preference relation on solution configurations. This allows coevolutionary algorithms to be naturally classified by the type of solution they seek. Using the weak preference relation also permits us to present a simpler definition of performance metrics than that used in previous coevolutionary No-Free-Lunch work, more akin to the definition used in the original No-Free-Lunch theorem.

The framework presented in this paper can be viewed as the combination of the ideas and definitions from two separate theoretical frameworks for analyzing search algorithms and coevolution consistent with the terminology of both.

We also present a new instance of free lunches in coevolution which demonstrates the applicability of our framework to analyzing coevolutionary algorithms based upon the solution concept which they implement.

## Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization; F.2 [**Analysis of Algorithms and Problem Complexity**]; I.2.8 [**Problem Solving, Control Methods, and Search**]

## General Terms

Theory

## Keywords

No Free Lunch, Coevolution, Solution Concept


*GECCO'08,* July 12–16, 2008, Atlanta, Georgia, USA.


## 1. INTRODUCTION

The No-Free-Lunch (NFL) theorem is a fundemental result in the field of function optimization [11]. In its most basic, and informal, form it states that all search algorithms perform equally well when averaged over all functions to be optimized. Such results provide a fascinating view into the underlying nature of black-box function optimization.

The original NFL theorem is applicable only to the class of iterative, data driven search algorithms which optimize an objective function. Due to the nature of fitness evaluation in coevolution, coevolutionary algorithms (CoEAs) are not, in general, a member of this class [12].

Recent work has shown that coevolution can exhibit free lunches [12]. That is, there are CoEAs which perform better than others when averaged over all interaction functions, with respect to some measure of performance. However, the question as to which classes of coevolution exhibit free lunches is still open.

The primary contribution of this paper is a novel framework for analyzing NFL like results for coevolution. While it has been shown that, in general, there are free lunches in coevolution, our framework allows NFL like inquiries to be phrased in terms of the type of solution desired. Our framework also has the advantage that it presents a standard and uniform view of CoEA performance comparison in terms of isomorphisms on labeled graphs induced by the search algorithms and the weak preference relation suggested in [4, 5]. This allows us to present a simpler definition of a performance metric for CoEAs than the one used in the proof of free lunches in coevolution [12], more akin to the definition used in the original NFL theorem [11].

The framework presented in this paper can be viewed as the combination of the ideas and definitions from two separate theoretical frameworks, one for analyzing search algorithms and the other for analyzing CoEAs, in a manner consistent with both. The framework we present is also capable of handling standard, non-coevolutionary, optimization.

The remainder of this paper is structured as follows. In Section 2 we provide a background of previous NFL frameworks for both traditional and coevolutionary optimization. Section 3 and Section 4 introduce solution concepts and the weak preference relation respectively, which we employ in our framework. In Section 5 we introduce our novel framework for analyzing NFL like results for coevolution. Section 6 shows how our framework may be employed in solution concept specific algorithm analysis. Conclusions and future lines of inquiry of this work are presented in Section 7.

## 2. NO-FREE-LUNCH BACKGROUND

In this section we provide an overview of previous NFL related work for both traditional function optimization as well as for coevolution. We also introduce the two models of search algorithms used in NFL work as well as in our framework.

### 2.1 Traditional Optimization

Here we present an overview of the framework used to prove the original NFL theorem [11].

Let $\mathcal{F} = \mathcal{Y}^{\mathcal{X}}$ be the set of $|\mathcal{Y}|^{|\mathcal{X}|}$ cost functions from a finite search space, $\mathcal{X}$, to the set of possible cost values, $\mathcal{Y}$.

A sequence of $m$ distinct cost function evaluations is denoted as:

$$d_m \equiv \{(d_m^x(1), d_m^y(1)), \cdots, (d_m^x(m), d_m^y(m))\}$$

where $d_m^x(i)$ is the $i$-th search space point visited and $d_m^y(i)$ is the value of the cost function evaluated at that point. Let $d_m^x$ and $d_m^y$ be the set of all the $m$ search points visited and their accompanying cost values, respectively. Also let $\mathcal{D}^*$ denote the set of all such finite sequences, $d_m$, including the empty sequence, $\emptyset$.

Informally, a search algorithm, such as an evolutionary algorithm, simulated annealing, and gradient descent, samples elements from the search space, evaluates the fitness of the sampled points and then selects new search space points based upon the previously seen points [11].

Formally, a search algorithm or a search heuristic, is defined as a function which takes as an argument a sequence of data points, pairs of search space points and their corresponding cost values, and outputs a new, unique, search space point [11]:

$$a : d_m \in \mathcal{D}^* \rightarrow \{x | x \notin d_m^x\} \tag{1}$$

where $x$ is an element of the search space, $\mathcal{X}$, which has not previously been visited. This is the first search algorithm model employed in our framework and we refer to it as the *traditional model*.

For simplicity we consider in this paper a search algorithm to be completely deterministic[1]. We refer the reader to [11, 12] for details on how to extend this to conditional probability distributions over the search space, where the next chosen point is a random variable in a probability distribution conditional upon the observed sequence.

#### 2.1.1 Theorem Statement

The NFL theorem says that the probability of seeing any sequence of cost values is constant when averaged uniformly over all optimization functions, independent of the algorithm used[2].

THEOREM 2.1 (NO FREE LUNCH). *For any two search algorithms $a$ and $b$, any positive integer $m$ and any sequence of $m$ cost values $d_m^y$:*

$$\sum_{f \in \mathcal{F}} P(d_m^y | f, m, a) = \sum_{f \in \mathcal{F}} P(d_m^y | f, m, b)$$

This result has been generalized to subsets of $\mathcal{F}$ which are closed under permutation, rather than $\mathcal{F}$ as a whole [7].

A search algorithm's performance is based upon the sequence of cost values it observes. Formally, a performance metric is a function mapping a sequence of costs values to a real number indicating how well an algorithm, which observed that sequence of cost values, performed:

$$\Phi : d_m^y \in D^* \rightarrow \Re$$

For example, a performance metric might be the last observed cost value, or the greatest cost value seen so far.

Since every sequence of cost values occurs the same number of times over all cost functions, regardless of the search algorithm, if follows that all search algorithms perform identically when averaged over all cost functions under any performance measure.

COROLLARY 2.2 (NO FREE LUNCH). *For any search algorithms $a$ and $b$, any positive integer $m$, any sequence of $m$ cost values $d_m^y$ and any performance metric $\Phi$:*

$$\sum_{f \in \mathcal{F}} P(\Phi(d_m^y) | f, m, a) = \sum_{f \in \mathcal{F}} P(\Phi(d_m^y) | f, m, b)$$

### 2.2 Coevolutionary Free Lunches

Free lunches exist in general in coevolution [12]. The setting in which coevolution was shown to exhibit free lunches was that of training an individual to compete in a multiplayer game, where the individual which maximized its worst case performance against all opponents was searched for. Under such conditions it was shown that there exist two algorithms with different performance when averaged over all interaction functions.

To do such an analysis, the framework used in the proof of the original NFL theorem was extended to the generalized optimization (GO) framework. This extension took place in three places.

First, the search space was expanded to include the set of individuals from all populations: $\mathcal{X} = P_1 \times \cdots \times P_n$. In the example used to show free lunches there were two distinct players ($\mathcal{X} = P_1 \times P_2$).

The model of a search algorithm was expanded to include both a search heuristic (Equation 1), which explores the interaction function, and a champion selection function, which selects a champion individual based upon the data points visited by the search heuristic.

Thus the search heuristic served the role of providing information about iteractions between players from all populations to the champion selection function, which then selected a champion from the population of interest believed to have the best worst case performance. The performance of the search algorithm as a whole was then based upon the quality of the selected champion after $m$ iterations of the search heuristic.

The search heuristic was identical to the search algorithm model used in the original NFL proof. The additional champion selection function was defined by:

$$A : D^* \rightarrow P_1 \tag{2}$$

where $P_1$ is the set of individuals, or strategies, for the player which the algorithm is designed to train. A generalization of the champion selection function (Equation 2) is the second search algorithm model employed in our framework and is referred to as the *candidate selection model*.

[1]Instances of stochastic algorithms may be viewed as deterministic when used with a specific pseudo-random number generator with a given seed [11].

[2]We leave the statement of the original NFL theorem in terms of probabilities even though we consider only deterministic search algorithms in our analysis.

Also the notion of a performance metric was expanded. To determine the performance of a given search algorithm, the worst case value of the selected champion must be considered, which requires dependence upon the cost, or interaction function, used. Thus to incorporate coevolution, the definition of a performance metric was expanded to allow the use of functions which depend on the interaction function. Formally the performance metric used in [12] to show free lunches in CoEAs was:

$$\Phi(C) = \min_{O \in P_2} g(C, O)$$

where $C$ is the champion individual from population $P_1$ selected by the algorithm, and $O$ ranges over the set of all opponents to $C$. This metric may be generalized to arbitrary $n$ player games by taking the minimum over all combinations of opponents.

Under such conditions it was shown that there is a pair of search algorithms with different performance, when averaged over all interaction functions. The reason for this was stated to be because a sum over all possible interaction functions $g : P_1 \times P_2 \to \mathcal{O}$, where $\mathcal{O}$ is an ordered set, is not equivalent to a sum over all functions $\min_{O \in P_2} g(C, O)$ [12].

Before introducing our novel framework for analyzing NFL like results for coevolution, we describe the notion of a solution concept and the weak preference relation [4] which we employ in our framework.

# 3. SOLUTION CONCEPTS

Solutions in coevolution take on various forms. For example, they may be probability distributions over sets of behaviors (as in the Nash equilibrium [4]) or may simply be an $n$-tuple of individuals (as in cooperative coevolution). Since the form of the solution in coevolution is dependent upon the problem at hand, in general CoEAs search for configurations over individuals from the evolving populations [1]. Both the form of these solution configurations as well as which such configurations are to be considered solutions is dependent upon the type of problem the CoEA is designed to solve, or the solution concept which it implements [4]. Thus solution concepts define the problem at hand and serve two purposes: they define the set of solution configurations as well as partition that set into a set of solutions and a set of non-solutions.

For discussing solution concepts we will adopt much of the notation from [1]. For a given solution concept and populations $P_1 \cdots P_n$ the set of possible solution configurations is denoted as $C(P_1 + \cdots + P_n)$. The actual form of the members of $C(P_1 + \cdots + P_n)$ is dependent upon the solution concept used. The goal of a CoEA is then to find a solution configuration which is considered a solution, as defined by the solution concept which it implements.

Recent theoretical work has focused on analyzing CoEAs in terms of the solution concept which they implement [4, 5]. The importantance of understanding and correctly implementing the desired solution concept has been shown, and pathologies often seen in CoEAs have been suggested to be a result of incorrect implementation of the desired solution concept [4]. However, the solution concepts themselves were the objects of interest, not the algorithms designed to implement, or at least approximate, them. In our work we approach this from the opposite direction and compare the performance of algorithms which implement the same solution concept. The framework we propose allows discussions of NFL like results for particular solution concepts. This allows the question of which classes of coevolution exhibit free lunches to be answered in terms of solution concepts, where the solution concepts take on the roles of the classes of coevolution.

Several different solution concepts have been used in coevolution, and much recent work has focused on designing CoEAs for specific solution concepts [9, 3]. Here we describe a few of them in order to illustrate the variety of forms which solution configurations may take.

## 3.1 Cooperative Coevolution

In many instances an optimization problem can naturally be decomposed into $n$ separate subproblems, $P_1, \cdots, P_n$, where candidate solutions to each subproblem are evolved simultaneously. In such cases the combination of candidate solutions which performs best with respect to some metric, $g$, is desired. Thus $C(P_1 + \cdots + P_n) = P_1 \times \cdots \times P_n$.

Formally, under this solution concept the set of solutions, from $P_1 \times \cdots \times P_n$, is defined by:

$$S = \{C : \forall C' \; g(C) \geq g(C')\}$$

This solution concept most closely resembles traditional function optimization in that the combination of subproblem candidate solutions which maximizes some objective function is desired.

## 3.2 Maximization Over All Test Cases

In many cases, coevolution is used to evolve candidate solutions which are best able to defeat, or solve, a set of test cases. As an example, consider evolving sorting networks and the accompanying sequences of numbers to sort [6].

The solution concept of maximization over all test cases requires a solution to maximize the outcome over all possible test cases. Formally there is a set of candidate solutions, $\mathcal{C}$, and a set of test cases, $\mathcal{T}$. Given an interaction function, $g : \mathcal{C} \times \mathcal{T} \to \mathcal{O}$, where $\mathcal{O}$ is an ordered set which determines the outcome of candidate solution $C$ on test $T$, the solution concept is defined by:

$$S = \{C \in \mathcal{C} : \forall C' \in \mathcal{C} \;\; \forall T \in \mathcal{T} \;\; g(C, T) \geq g(C', T)\}$$

Thus, $C(\mathcal{C} + \mathcal{T}) = \mathcal{C}$.

Theoretical aspects of this solution concept were discussed in [10]; however, in many real-world problems there are no candidate solutions which perform best over all possible test cases. That is, there is often a trade-off between performance on test cases. In such situations the candidate solutions which are expected to perform best against a random test case might be desired as described in the next solution concept.

## 3.3 Maximization of Expected Utility

In this solution concept there is also a set of candidate solutions, $\mathcal{C}$, and a set of test cases, $\mathcal{T}$. An element of $\mathcal{C}$ is a solution if and only if it maximizes the expected outcome of a uniform randomly selected test case. Formally the solution set in this solution concept is defined by:

$$S = \{C \in \mathcal{C} : \forall C' \in \mathcal{C} \;\; E(g(C, T)) \geq E(g(C', T))\}$$

where $E$ is the expectation operator. In the simple case where a candidate solution either passes or fails the tests,

this is equivalent to maximizing the number of tests the candidate solution passes.

This solution concept, or a derivative with a non-uniform distribution of test cases, is one of the most commonly used in coevolution literature, and is often stated as the canonical example of competitive coevolution, where test cases are evolved to challenge the current candidate solutions and thus force them to improve in quality.

## 3.4 Nash Equilibrium

Game theory provides the concept of the Nash equilibrium solution concept [4]. A Nash equilibrium in an $n$ player game is a specification of a strategy for each player such that no single player can profit by a change in that player's strategy alone. Thus for any player to profit, two or more players must cooperate.

Formally, given $n$ players, where each player has a corresponding set of possible pure strategies or behaviors $B_i$, a Nash equilbrium is a mixed strategy for each player, where a mixed strategy for player $i$ is a probability distribution over the set of behaviors in $B_i$. Let $\Delta B_i$ denote the set of probability distributions over the set of behaviors $B_i$. A member, $\alpha = (\alpha_1, \cdots, \alpha_n)$, of the set $\Delta B_1 \times \cdots \times \Delta B_n$ is a Nash equilbrium if and only if for all players $i$ and all $\beta_i \in \Delta B_i$ $E(g_i(\alpha)) \geq E(g_i(\alpha_1, \cdots, \alpha_{i-1}, \beta_i, \alpha_{i+1}, \cdots, \alpha_n))$, where $g_i$ denotes the payoff for player $i$.

Thus the solution set for this solution concept is given by:

$$S = \{\alpha \in \Delta B_1 \times \cdots \times \Delta B_n : \forall i \, \forall \beta_i \in \Delta B_i$$
$$E(g_i(\alpha)) \geq E(g_i(\alpha_1, \cdots, \alpha_{i-1}, \beta_i, \alpha_{i+1}, \cdots, \alpha_n))\}$$

and the set of solution configurations is $\Delta B_1 \times \cdots \times \Delta B_n$.

## 3.5 Maximin

This solution concept also has a set of candidate solutions (solution configurations), $\mathcal{C}$, and test cases, $\mathcal{T}$. The solutions are those solution configurations which maximize the minimum outcome over all test cases. This is the solution concept which was shown to exhibit free lunches under the GO framework [12].

The solution set is given by:

$$S = \{C \in \mathcal{C} : \forall C' \in \mathcal{C} \ \min_{T \in \mathcal{T}}(g(C,T)) \geq \min_{T \in \mathcal{T}}(g(C',T))\}$$

# 4. COEVOLUTIONARY PROGRESS

Assessing CoEA progress is a difficult problem and requires consideration of the solution concept being employed [2, 8, 5]. A generic, solution concept independent, weak preference relation was suggested and used in [5] to analyze the theoretical performance of CoEAs with unbounded memory under various solution concepts.

We employ this weak preference relation in our framework to compare the performance of particular instances of CoEAs.

## 4.1 Weak Preference

The solution set defined by a particular solution concept depends upon the interaction function under consideration and the context in which individuals are evaluated, that is the set of other individuals under consideration [5]. Consider the maximization of expected utility solution concept with candidate solution set $\mathcal{C}$ and test case set $\mathcal{T}$. A given candidate solution may appear in the solution set when compared against a subset of the other candidate solutions and test cases. That is, it may appear in the solution set defined on $C \subset \mathcal{C}$, $T \subset \mathcal{T}$, but may not be a member of the solution set defined on the problem instance as a whole, for example by judging it against only those tests on which it performs optimally. A solution set context, which for brevity's sake we will further refer to simply as a context, is defined to be the subset of each population which is currently under consideration. Given a set $X = p_1 + \cdots + p_n$ where $p_i \subseteq P_i$, the solution set defined by that context is denoted $\delta X$.

The weak preference relation is a binary relation on solution configurations [5]. Formally, a candidate solution configuration, $\alpha$, is weakly preferred to a candidate solution configuration, $\beta$, written $\alpha \succ \beta$, if every context in which $\beta$ appears as a solution is a strict subset of a context in which $\alpha$ is a solution.

Definition 4.1 (Weak Preference). *A solution configuration $\alpha$ is weakly preferred to a solution configuration $\beta$, written $\alpha \succ \beta$, iff for every context $X_\beta$ with $\beta \in \delta X_\beta$ there is a context $X_\alpha$ such that $X_\beta \subset X_\alpha$ and $\alpha \in \delta X_\alpha$.*

Thus any solution is preferred to any non-solution, as desired.

The weak preference relation has been shown to be irreflexive, asymmetric and transitive [5], and can be viewed as a type of order on the set of configurations[3].

# 5. FRAMEWORK

We approach the question of the existence of free lunches in CoEAs from the point of view of solution concepts and the weak preference relation. Informally, we say that a given solution concept exhibits no free lunches if for every pair of algorithms, $a$ and $b$, and every interaction function $g$, there is a corresponding interaction function $g'$ such that the labeled graph induced by the weak preference relation and algorithm $a$ on $g$ is isomorphic to the labeled graph induced by $b$ on $g'$. That is, there is no way to distinguish the performance of algorithm $a$ on $g$, under the weak preference relation, from the performance of $b$ on $g'$, up to isomorphism.

We begin the formalization of our framework by defining how a solution concept and the weak preference relation induce a directed acyclic graph (DAG) on the set of possible solution configurations. We then show how search algorithms may be viewed in terms of labeling nodes in the preference DAG.

## 5.1 Preference DAG

The weak preference relation induces a directed graph on the space of solution configurations, $C(P_1 + \cdots + P_n)$. We define an edge to be from configuration $\beta$ to configuration $\alpha$ when $\alpha \succ \beta$. That is, edges point in the direction of increased preference and may be thought of as routes through configuration space to the global solutions. We desire algorithms which are capable of quickly climbing the preference graph. As the weak preference relation is irreflexive, asymmetric and transitive, it follows that the preference directed graph is always acyclic (i.e., a DAG).

Given a solution concept, each interaction function induces a different preference DAG. If functions $g$ and $g'$ induce an isomorphic preference DAG we will say that $g$ and $g'$ are isomorphic.

[3]If we were to define $\alpha \succ \alpha$ for all solution configurations $\alpha$ (the reflexive closure of $\succ$) then the weak preference relation would be a partial order on the set of configurations.

## 5.2 Algorithm Models

We show how both search algorithm models used in NFL literature are utilized in our framework, as well as how they both relate to the preference DAGs.

### 5.2.1 Traditional Model

Depending on the solution concept of interest, the solution configurations (nodes in the preference DAG) may themselves be the objects used as arguments to the interaction function, but in general this need not be the case. That is, in general a candidate solution configuration need not be a member of $P_1 \times \cdots \times P_n$, see for example the Nash equilibrium; or in other words $C(P_1 + \cdots + P_n)$ need not equal $P_1 \times \cdots \times P_n$.

In the simplest case, where $C(P_1 + \cdots + P_n) = P_1 \times \cdots \times P_n$, the search algorithm model used in the original NFL proof (see Equation 1), can be used. In such cases the points visited by the algorithm can be viewed as a direct traversal of the nodes in the preference DAG.

Formally, say a given algorithm explores the sequence of $m$ data points $(x_1, y_1), \cdots, (x_m, y_m)$ (by exploration we mean examination under the interaction function, i.e., fitness evaluations). We can view the visiting of the point $x_i$ as labeling that configuration in the preference DAG with the label $i$. Thus after $m$ iterations of some search algorithm, $m$ nodes in the preference DAG have been labeled, indicating at which point during the search they were examined, or visited.

### 5.2.2 Candidate Selection Model

In the most general case where the solution configurations are not simply $n$-tuples of members from each population, a search algorithm model similar to the one presented in [12] must be used, where a search heuristic explores the interaction function and a candidate selection function maps a set of previously visited data points explored by the search heuristic to a solution configuration.

Formally, a candidate selection function is of the form:

$$A : D^* \rightarrow C(P_1 + \cdots + P_n) \tag{3}$$

This is a generalization of the champion selection function, Equation 2 [12]. We further assume that the candidate selection function does not depend upon the order of the data points in the sequence $d_m$, and does not depend upon the "names" of the individuals. That is, if $d_m$ and $d'_m$ are two permutations of the same set of data points, then $A(d_m) = A(d'_m)$ for all candidate selection functions $A$. Also, given any sequence $d_m$ (on populations $P_1 \cdots P_n$), if all occurrences of the individuals $p_1, p_2 \in P_i$ in $d^x_m$ were swapped to form a new sequence $d'_m$, but the observed cost values remained the same, then the solution configuration returned by $A(d'_m)$ would be identical to that of $A(d_m)$ except the roles of $p_1$ and $p_2$ would be swapped.

The candidate selection function can also be viewed as a memory mechanism, or archive, in a sense similar to that used in [4], in that the memory mechanism, or candidate selection function, has the role of representing the solution, relieving the search heuristic of that responsibility. In such cases the search heuristic's sole responsibility is to provide relevant information to the candidate selection function.

While we can imagine comparing the performance of arbitrary combinations of search heuristics and candidate selection functions, we restrict our attention to a fixed, but arbitrary, candidate selection function. We do this because allowing arbitrary combinations, or even a fixed, but arbitrary, search heurisitic combined with different candidate selection functions will always exhibit free lunches. To see this consider an arbitrary solution set, $S$, and one candidate selection function which when presented with all possible data points never selects a member of the solution set and a second selection function which always does. When we let the number of data points, $m$, be the total number of combinations of behaviors from each population, then, for all interaction functions, the second selection function will always outperform the first, regardless of the search heuristic.

Before presenting definitions of when No-Free-Lunches occur in our framework, we first define the notion of a performance metric for our framework.

## 5.3 Performance Metrics

The weak preference relation (Definition 4.1) provides a means by which to measure performance in coevolution. Algorithms which consistently find more preferred solution configurations than other algorithms are desired. Performance is thus a measure of how desirable a returned solution configuration is, with respect to the weak preference relation.

Formally, a performance metric, in our framework, is defined to be a function $\Phi$ from the set of labeled DAGs to the real numbers, where each node's label is a natural number such that no natural numbers are skipped. That is, if the number $m$ appears in the graph as a label, then so do all positive integers $n < m$. Thus the performance of an arbitrary algorithm depends only upon the shape and labeling of the preference DAG. It follows that for any performance metric, $\Phi$, if two labeled graphs $g$ and $g'$ are isomorphic, then $\Phi(g) = \Phi(g')$. We exploit this fact in the No-Free-Lunch definitions given next.

An example of a performance metric might be the number of solution configurations which the last labeled node is weakly preferred to.

Intuitively this definition makes sense as we care only about the relative locations of the solution configurations selected (or visited) by the search algorithm in the preference DAG and not about the "names" of the particular configurations.

The possible dependence upon the interaction function is removed from the performance metric in our framework, thus allowing a simpler definition, more akin to that of the definition in the original NFL framework. This dependence is instead hidden in the notion of a solution concept and the weak preference relation.

## 5.4 No-Free-Lunch Definitions

In this section we present our definitions of when no free lunches occur in our framework for both search algorithm models, that is with and without a candidate selection function. Informally, the NFL theorem applies to a solution concept only when for all search algorithm pairs $a$ and $b$, any performance metric value, $V$, and any performance metric $\Phi$, $\Phi$ takes on the value $V$ under search algorithm $a$ for the same number of interaction functions as it does under algorithm $b$. As noted, this occurs when for every labeled graph induced by algorithm $a$ on an interaction function $g$ there is an isomorphic labeled graph induced by $b$ on a corresponding interaction function $g'$.

### 5.4.1 Traditional Model

If given a solution concept of interest, two search algorithms $a$ and $b$ on interaction functions $g$ and $g'$, respectively, produce labeled graphs which are isomorphic to one another, then the performance of $a$ on $g$ and $b$ on $g'$ are indistinguishable under any performance metric.

Definition 5.1 (Traditional Model - NFL). *A solution concept is said to not exhibit free lunches if for any pair of algorithms $a$ and $b$ and any positive integer $m$, there is a bijection $\mathcal{F} : \mathcal{G} \rightarrow \mathcal{G}$, where $\mathcal{G}$ denotes the set of all interaction functions from the search space to the space of possible outcomes, such that:*

$$\forall g \in \mathcal{G} \;\; a_g^m(\emptyset) \simeq b_{\mathcal{F}(g)}^m(\emptyset)$$

*where $a_g^m(\emptyset)$ denotes the labeled graph induced by algorithm $a$ after $m$ iterations on interaction function $g$, and $\simeq$ is the isomorphism relation[4].*

For the remainder of this paper we use the name of an algorithm $a$ to denote both the function mapping data points to a new search space point (as in Equation 1) as well as the graph induced by such a mapping. The intended usage should be clear from the context.

In other words, Definition 5.1 says that $\mathcal{F}$ is a one-to-one correspondence such that the graph induced by $a$ under $g$ is isomorphic to the graph induced by $b$ under $\mathcal{F}(g)$. Thus for any performance metric, $\Phi$, it follows that:

$$\forall g \in \mathcal{G} \Phi(a_g^m(\emptyset)) = \Phi(b_{\mathcal{F}(g)}^m(\emptyset))$$

So, every possible performance value occurs the same number of times for any pair of algorithms when ranged over all possible interaction functions.

### 5.4.2 Candidate Selection Model

For the more general search algorithm, we consider just the solution configuration produced after $m$ iterations of the search heuristic, line was done in [12]. As with the traditional model, we consider the solution configuration returned by the candidate selection function to be equivalent to labeling the corresponding node in the preference DAG. Again, two algorithms run on two interaction functions are then considered to have indistinguishable performance after $m$ iterations if and only if the labeled graphs produced by both are isomorphic.

For the candidate selection algorithm model we provide two versions of our NFL theorem. The weak version considers only a single candidate selection function, while the strong version considers every possible selection function.

Definition 5.2 (Candidate Model - Weak NFL). *A solution concept and candidate selection function, $A$, combination is said to not exhibit free lunches if for any pair of search heuristics $a$ and $b$, and any positive integer $m$, there is a bijection $\mathcal{F} : \mathcal{G} \rightarrow \mathcal{G}$ such that:*

$$\forall g \in \mathcal{G} \;\; A(a_g^m(\emptyset)) \simeq A(b_{\mathcal{F}(g)}^m(\emptyset)) \tag{4}$$

*where $A(a_g^m(\emptyset))$ denotes the graph induced by the interaction function $g$ with the single node selected by candidate selection function, $A$, labeled as being selected.*

[4]In reality this depends also upon the solution concept used; however, we will largely leave this relationship implicit.

As with the traditional model, we will use $A(d_m)$ to represent both the particular solution configuration returned by the candidate selection function, as well as the labeled graph induced by the algorithm.

Unlike the traditional optimization algorithm case, the labeled graphs will always consist of a single label regardless of the number of iterations performed by the search heuristic.

Definition 5.3 (Candidate Model - Strong NFL). *A solution concept is said to not exhibit free lunches if for any candidate selection function, $A$, any pair of search heuristics $a$ and $b$, and any positive integer $m$, there is a bijection $\mathcal{F} : \mathcal{G} \rightarrow \mathcal{G}$ such that Equation 4 is satisfied.*

Thus the Strong NFL holds for a solution concept only when the Weak NFL holds for that solution concept and any candidate selection function.

Note that we do not require that the bijection $\mathcal{F}$, for all values of $m$, produces interaction functions on which algorithm $b$ produces an isomorphic graph to that of $a$. $\mathcal{F}$ may be a function of $m$, thus different values of $m$ may very well produce different bijections.

As in the case of the traditional model, given such a bijection $\mathcal{F}$ it follows that:

$$\forall g \in \mathcal{G} \Phi(A(a_g^m(\emptyset))) = \Phi(A(b_{\mathcal{F}(g)}^m(\emptyset)))$$

for any performance metric $\Phi$.

The NFL definitions for the candidate selection model deal with the informativeness of search heuristics. That is, if a solution concept and candidate selection function satisfy Definition 5.2, then all search heuristics provide equally relevant information to the candidate selection function, when averaged over all interaction functions. In other words, no particular search heuristic consistently provides more relevant information to the candidate selection function than any other heuristic.

## 6. ANALYSIS

We now show that the maximization over all test cases solution concept exhibits free lunches under the weak preference relation.

Theorem 6.1 (Maximization Over All Test Cases). *The solution concept of maximization over all test cases along with the Bayes optimal rule candidate selection function, which always selects the candidate solution with the greatest expected value, averaged over all interaction functions consistent with the observed sample, exhibits free lunches under the weak preference relation. That is, there is a pair of algorithms $a$ and $b$ and positive integer $m$ such that there is no bijection $\mathcal{F}$ which satisfies:*

$$\forall g \in \mathcal{G} \;\; A(a_g^m(\emptyset)) \simeq A(b_{\mathcal{F}(g)}^m(\emptyset))$$

*where $A$ is the Bayes optimal rule candidate selection function, and $\mathcal{G}$ is the set of all interaction functions.*

Proof. Let $\mathcal{C}$ be a finite set of candidate solutions, $\mathcal{T}$ be a finite set of test cases and $\mathcal{Y}$ be a finite set of cost values. For simplicity assume that $|\mathcal{C}| = |\mathcal{T}| = m$ and $\mathcal{Y} = \{1, 2\}$. An assignment of 1 to the candidate solution $c$, test case $t$ pair can be thought of as $c$ failing the test $t$ and an assignment of 2 can be thought of as $c$ passing $t$.

As was done in [12], algorithm $a$ explores the candidate selection/test case combinations $(c_1, t_1), (c_2, t_1), \cdots, (c_m, t_1)$

and algorithm $b$ explores the candidate selection/test case combinations $(c_1, t_1), (c_1, t_2), \cdots, (c_1, t_m)$. We will show that algorithm $a$ returns a member of the solution set (i.e., a candidate solution which performs no worse than any other candidate solution on any test case) on a greater number of interaction functions then does $b$, which will imply that no bijection $\mathcal{F}$ exists which satisfies

$$\forall g \in \mathcal{G} \quad A(a_g^m(\emptyset)) \simeq A(b_{\mathcal{F}(g)}^m(\emptyset))$$

as the existence of such an $\mathcal{F}$ implies that both algorithms return the same number of members of the solution set, over all interaction functions.

First consider randomly selecting a candidate solution $c$. $c$ is a member of the solution set, under an interaction function $g$, if and only if for all test cases $t$ and all candidate solutions $c'$ $g(c,t) \geq g(c',t)$. Thus, if $c$ receives a value of 1 on a test case $t$ (i.e., fails $t$) then so must all other candidate solutions $c'$, for $c$ to appear in the solution set. Assume that $c$ passes the first $k$ test cases (receives 2's on the first $k$ test cases) and fails the other $m-k$ test cases. There are then $2^{(m-1)m}$ functions consistent with the observed sample where $c$ passes the first $k$ test cases and fails the remaining $m-k$ (all possible assignments of outcomes to the remaining $(m-1)m$ candidate solution/test case pairs). Of those functions, $c$ appears in the solution set only in those where all candidate solutions fail the last $m-k$ test cases. In functions in which $c$ is a solution, there are $(m-1)k$ candidate solution/test case pairs for which there are two possible outcomes and $(m-1)(m-k)$ remaining pairs for which there is only one possible outcome. Thus, $c$ is a member of the solution set under $2^{(m-1)k}$ interaction functions whenever $c$ passes exactly $k$ tests. There are $\binom{m}{k}$ ways in which $c$ can pass exactly $k$ test cases. Summing over the number of test cases which $c$ passes yields

$$\sum_{k=0}^{m} \binom{m}{k} 2^{(m-1)k}$$

many interaction functions in which $c$ is a member of the solution set. Which by the binomial theorem is

$$(2^{m-1}+1)^m$$

Thus, every candidate solution appears in the solution set in exactly $(2^{m-1}+1)^m$ interaction functions.

Now consider algorithm $a$ which explores the sequence of candidate selection/test case combinations $(c_1, t_1), \cdots, (c_m, t_1)$. The Bayes optimal rule candidate selection function selects a candidate solution which performs best against $t_1$. Assume, without loss of generality, that $c_1$ performs no worse than any other candidate solution on test $t_1$, and algorithm $a$ selects $c_1$. Since no other candidate solution outperforms $c_1$ on $t_1$, $c_1$ is a member of the solution set only when no other candidate solution passes one of the tests $t_2, \cdots, t_m$ when $c_1$ does not. Thus, the analysis used above for a randomly selected candidate solution may be applied here as well, except there are now $m-1$ test cases instead of $m$. The selected candidate solution then appears in the solution set of $(2^{m-1}+1)^{m-1}$ of the interaction functions consistent with the observed sample (i.e., $(c_1, t_1), (c_2, t_1), \cdots, (c_m, t_1)$). This is true for all possible observed samples, and there are $2^m$ possible samples resulting in

$$2^m \cdot (2^{m-1}+1)^{m-1}$$

interaction functions under which algorithm $a$ returns a member of the solution set.

Algorithm $b$ explores the sequence, $(c_1, t_1), \cdots, (c_1, t_m)$, of candidate solution/test case combinations. The Bayes optimal rule candidate selection function selects either $c_1$ or another randomly selected candidate solution, $c_i$, depending on which one is expected to appear in the solution set under more interaction functions consistent with the observed sample.

Let $k$ be the number of test cases which $c_1$ passes. $c_1$ is then a member of the solution set in $2^{(m-1)k}$ of the interaction functions consistent with the observed sample. Now consider in how many interaction functions consistent with the observed sample the randomly selected candidate solution, $c_i$, will appear as a solution. $c_i$ will appear in the solution set only when it passes all tests which $c_1$ does. Since $c_1$ passes $k$ tests, $c_i$ must also pass those same $k$ tests and may either pass or fail the remaining $m-k$ tests. Let $j$ be the number of additional tests that $c_i$ passes, then $c_i$ is a member of the solution set whenever the other remaining candidate solutions fail the remaining $m-k-j$ tests and either pass or fail the $k+j$ tests. Summing over the number of additional test cases which $c_i$ solves yields

$$\sum_{j=0}^{m-k} \binom{m-k}{j} 2^{(m-2)(k+j)}$$

interaction functions consistent with the observed sample under which $c_i$ is a member of the solution set. Which by the binomial theorem is equal to

$$2^{(m-2)k} \cdot (2^{m-2}+1)^{m-k}$$

Algorithm $b$ then selects either $c_1$ or $c_i$, depending on which is a member of the solution set in more interaction functions consistent with the observed sample. Summing over the number of test cases which $c_1$ passes yields

$$\sum_{k=0}^{m} \binom{m}{k} \max\left(2^{(m-1)k}, 2^{(m-2)k} \cdot (2^{m-2}+1)^{m-k}\right)$$

Which is equal to

$$\sum_{k=0}^{m} \binom{m}{k} 2^{(m-2)k} \cdot \max\left(2^k, (2^{m-2}+1)^{m-k}\right)$$

By inspection it is easy to see that $2^k$ is greater than $(2^{m-2}+1)^{m-k}$ (i.e., algorithm $b$ selects $c_1$) only when $k$ is $m$ or $m-1$. Thus, the above sum may be rewritten as

$$\sum_{k=0}^{m} \left( \binom{m}{k} 2^{(m-2)k} \cdot (2^{m-2}+1)^{m-k} \right) + \binom{m}{m} 2^{m(m-2)}(2^m - 1) + \binom{m}{m-1} 2^{(m-1)(m-2)}(2^{m-1} - 2^{m-2} - 1)$$

Which may be simplified to

$$(2^{m-1}+1)^m + 2^{m(m-2)}(2^m - 1) + m \cdot 2^{(m-1)(m-2)} \cdot (2^{m-1} - 2^{m-2} - 1)$$

Figure 1 shows the percentage of interaction functions in which algorithms $a$ and $b$ select members of the solution

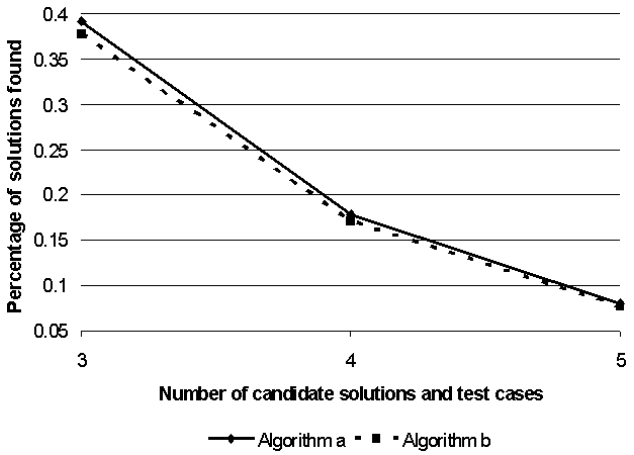


**Figure 1: Algorithm Performance.**

set, for small values of $m$. As can be seen, the number of interaction functions under which $a$ and $b$ select members of the solution set is different, and therefore no such bijection $\mathcal{F}$ exists. □

# 7. CONCLUSIONS & FUTURE WORK

The primary contribution of this paper is a novel framework for analyzing NFL like results for coevolution. Our framework combines the ideas from the framework used to prove the original NFL theorem and the framework used to analyze the theoretical performance of CoEAs in terms of the solution concept they implement presented in [4].

Our framework employs the weak preference relation to measure and compare the performance of different algorithms, and the question of the existence of free lunches is phrased in terms of isomorphisms on this relation. This allows us to present a simpler definition of a performance metric than used in previous coevolutionary NFL work, more akin to the definition used in the original NFL framework. This is possible because the dependence on the interaction function, often found in coevolution, is hidden in the notion of a solution concept and the weak preference relation.

The use of the weak preference relation also easily allows our framework to be used to discuss NFL like results in terms of the solution concept of interest. As it has been shown that in general coevolution does have free lunches, the natural next question is what classes of CoEAs possess free lunches. Our framework allows such questions to be addressed in terms of solution concepts.

We formalize both search algorithm models used in related NFL literature, for both traditional and coevolutionary optimization, in our framework. We also present a new instance of free lunches in coevolution which demonstrates the applicability of our framework to analyzing coevolutionary algorithms based upon the solution concept which they implement.

As there are many different solution concepts used throughout CoEA literature, the next step for the use of this framework is to apply it to analyzing a variety of different solution concepts. This work also raises the question as to what properties a solution concept must possess in order to exhibit (or not exhibit) free lunches. For example, the two different coevolutionary solution concepts shown to exhibit free lunches, one given in [12] and one in this paper, both are examples of candidate solution/test case coevolution; however, it is unknown whether or not all classes of candidate solution/test case coevolution exhibit free lunches. Our framework's ability to frame such NFL questions in terms of solution concepts provides a means by which to investigate such inquiries.

# 8. ACKNOWLEDGEMENTS

We would like to thank Kate Holdener for her valuable reviews of this paper.

# 9. REFERENCES

[1] Anthony Bucci and Jordan B. Pollack. Thoughts on Solution Concepts. In *Proceedings of the 9th annual Genetic and Evolutionary Computation Conference*, pages 434–439, New York, NY, USA, 2007. ACM.

[2] John Peter Cartlidge. *Rules of Engagement: Competitive Coevolutionary Dynamics in Computational Systems*. PhD thesis, University of Leeds, 2004.

[3] Edwin de Jong. The Maxsolve Algorithm for Coevolution. In *Proceedings of the 7th annual Genetic and Evolutionary Computation Conference*, pages 483–489, New York, NY, USA, 2005. ACM.

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

[5] Sevan G. Ficici. Monotonic Solution Concepts in Coevolution. In *Proceedings of the 7th annual Genetic and Evolutionary Computation Conference*, pages 499–506, New York, NY, USA, 2005. ACM.

[6] Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D Nonlinear Phenomena*, 42:228–234, June 1990.

[7] Christian Igel and Marc Toussaint. No-Free-Lunch Theorem for Non-Uniform Distributions of Target Functions. *Journal of Mathematical Modelling and Algorithms*, 3(4):1570–1166, Dec 2004.

[8] Edwin D. De Jong. Objective Fitness Correlation. In *Proceedings of the 9th annual Genetic and Evolutionary Computation Conference*, pages 440–447, New York, NY, USA, 2007. ACM.

[9] Frans A. Oliehoek, Edwin D. de Jong, and Nikos Vlassis. The Parallel Nash Memory for Asymmetric Games. In *Proceedings of the 8th annual Genetic and Evolutionary Computation Conference*, pages 337–344, New York, NY, USA, 2006. ACM.

[10] Christopher Darrell Rosin. *Coevolutionary Search Among Adversaries*. PhD thesis, University of California - San Diego, 1997.

[11] David Wolpert and William Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, Apr 1997.

[12] David Wolpert and William Macready. Coevolutionary Free Lunches. *IEEE Transactions on Evolutionary Computation*, 9(6):721–735, Dec 2005.