Co-optimization: a generalization of coevolution

Travis Service

CO-OPTIMIZATION: A GENERALIZATION OF COEVOLUTION

by

TRAVIS SERVICE

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

in

COMPUTER SCIENCE

2008

Approved by

Dr. Daniel Tauritz, Advisor
Dr. Bruce McMillin
Dr. David Grow

ABSTRACT

Many problems encountered in computer science are best stated in terms of interactions amongst individuals. For example, many problems are most naturally phrased in terms of finding a candidate solution which performs best against a set of test cases. In such situations, methods are needed to find candidate solutions which are expected to perform best over all test cases.

Coevolution holds the promise of addressing such problems by employing principles from biological evolution, where populations of candidate solutions and test cases are evolved over time to produce higher quality solutions. Coevolution has had both success stories as well as noted deficiencies, and many additions to the base coevolutionary algorithm have been proposed to address some of the studied deficiences; however, to the author's knowledge, all such additions have been based on and applied to an underlying evolutionary model. This thesis presents a generalization of coevolution to *Co-Optimization*, where optimization techniques that do not rely on evolutionary principles may be used. Instead of introducing a new addition to coevolution in order to make it better suited for a particular class of problems, this thesis suggests removing the evolutionary model in favor of a technique better suited for that class of problems.

This thesis makes three distinct contributions. The primary contribution is a generalization of coevolution to co-optimization. Co-optimization allows arbitrary black-box optimization techniques to be employed in interactive domains in place of artificial evolution. The second contribution is a methodology based on co-optimization for Critical Infrastructure Protection as well as a detailed real-world case study on the use of this methodology for strengthening the electric power transmission system. The third and final contribution is a theoretical framework for discussing No-Free-Lunch like results for co-optimization, and thus also for coevolution. Informally, the No-Free-Lunch theorem states that all black-box optimization techniques perform equally well when averaged over all functions to be optimized. The framework presented in this thesis allows such results to be proven for classes of co-optimization.

ACKNOWLEDGMENT

First, I would like to thank my advisor Dr. Daniel Tauritz for all the help and guidance he has provided as well as for introducing me to the field of evolutionary algorithms. I would also like to thank the other members of my committee: Dr. Bruce McMillin and Dr. David Grow for the assistance they have provided.

Ekaterina “Kate” Holdener’s reviews and comments on this thesis were invaluable and are much appreciated.

I would also like to thank my friends and family for the support and encouragement they have given me throughout this work.

Finally, I would like to thank the Power Group at the Missouri University of Science and Technology for funding the inital portions of this work.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

Many problems in computer science can be phrased in terms of searching through a set of possible solutions, or search space, for a candidate solution which best satisfies some criteria. Traditionally this involves optimizing an objective function, which assigns a measure of quality to each candidate solution in the search space. For example, the objective function could give the overhead cost associated with each element in the search space, in which case the minimum value is desired, or may give the expected profit for each member of the search space, in which case the maximum is desired.

While traditional function optimization naturally applies to a vast number of common problems, including difficult real-world problems such as from the field of Critical Infrastructure Protection [1], some problems are best phrased in terms of interactions amongst individuals. For example, many problems are most naturally phrased in terms of finding a candidate solution which performs best against a set of test cases. For large numbers of test cases it is often infeasible to evaluate a candidate solution against all test cases.

Coevolution holds the promise of addressing such problems by employing principles from biological evolution [2, 3]. Populations of individuals are evolved over a period of time to produce higher quality individuals. As an example, consider the case of finding a candidate solution which performs best against a set of test cases. In such situations two-population competitive coevolution can be employed to coevolve a population of candidate solutions along with a population of test cases. Evolutionary principles are used to evolve the candidate solutions such that they perform well against the evolving population of test cases, and to evolve the test cases so that they provide a challenge to the population of candidate solutions. In such a manner, evolutionary principles guide the populations toward higher quality candidate solutions and more challenging test cases.

Coevolution has had both success stories as well as noted deficiencies [4, 5]. Many additions to coevolution have been proposed to help address some of the studied deficiencies; however, to the author's knowledge, all such additions have been based on and applied to an underlying evolutionary model. This thesis presents a generalization of coevolution to *Co-Optimization*, where optimization techniques that do not rely on evolutionary principles may be used. Instead of introducing a new addition to coevolution in order to make it better suited for a particular class of problems, this thesis provides a generalized framework within which the most suitable optimization technique for a given problem can be employed.

1.1. TRADITIONAL FUNCTION OPTIMIZATION

Black-box function optimization techniques are a class of optimization algorithms where the objective function is treated as a black-box which takes an element of the search space and returns the associated objective value. As function optimization is central to many of the problems faced in computer science, a variety of black-box optimization algorithms exists. This thesis focuses its attention on four in particular which are presented here.

1.1.1. Gradient Ascent. Gradient ascent is a member of the class of hill climbing optimization algorithms in which successive steps are taken to neighboring points in the search space of higher value than the current point. In the continuous version this is accomplished by moving along the gradient of the function. In discrete cases, or cases where the gradient can not be computed, neighboring points may be selected at random and moved to if they are of a greater value then the current point.

1.1.2. Local Beam Search. Local beam search is a population based hill climbing optimization algorithm [6]. The algorithm maintains a collection of k, initially random, points in the search space. During each iteration of the algorithm a random successor of each of the k points is generated and the best k points are selected to progress to the next iteration. Local beam search is, in some sense, a population based version of gradient ascent.

1.1.3. Simulated Annealing. Simulated annealing is an optimization technique which employs principles from annealing in metallurgy [6, 7]. A model of temperature is used to incorporate a degree of stochasticity into the search, the higher the temperature the greater the chance of moving to a less desirable point in the search space. The system begins in a high temperature state and as the search progresses the temperature is gradually lowered, according to a cooling schedule.

The search is successively moved to neighboring points in the search space in a probabilistic fashion, with probability proportional to the difference between the objective value of the generated point and that of the current point. If the generated neighboring point is of higher value than the current point, then the search is moved to that generated point. Thus, over time the search progresses to areas of the search space with higher objective values.

1.1.4. Evolutionary Computation. Evolutionary computation applies principles from biological evolution to optimization problems [2, 3]. A population of individuals, containing search space points, are evolved over multiple generations. Individuals reproduce creating offspring, and then compete for survival within the population. Selection pressures guide the evolving population toward more promising areas of the search space. A fitness value is assigned to each individual in the population such that fitter individuals are better with respect to the objective funtion, and the fitter an individual is the more likely it will be able to reproduce as well as survive to future generations.

In traditional evolutionary computation, the fitness of an individual is determined by an external function (i.e., the function to be optimized).

1.2. COEVOLUTION

Coevolutionary optimization is an extension of traditional evolutionary computation in which individuals are evaluated based upon interactions with their peers rather than by an external fitness function [2, 3]. Figure 1.1 diagrams the two-population coevolutionary algorithm (CoEA). Both populations evolve in separate life cycles which intersect during fitness computation.

Coevolution has been successfully applied to optimization problems. For instance, in [4] sorting networks and sequences of numbers to be used as test cases were evolved simultaneously. The performance of coevolution is compared to that of evolution against random samples of test cases and better results are obtained through the use of coevolution.

While such examples highlight the potential of coevolution as a problem solving technique, there are several problems inherent to coevolution described in literature, including:

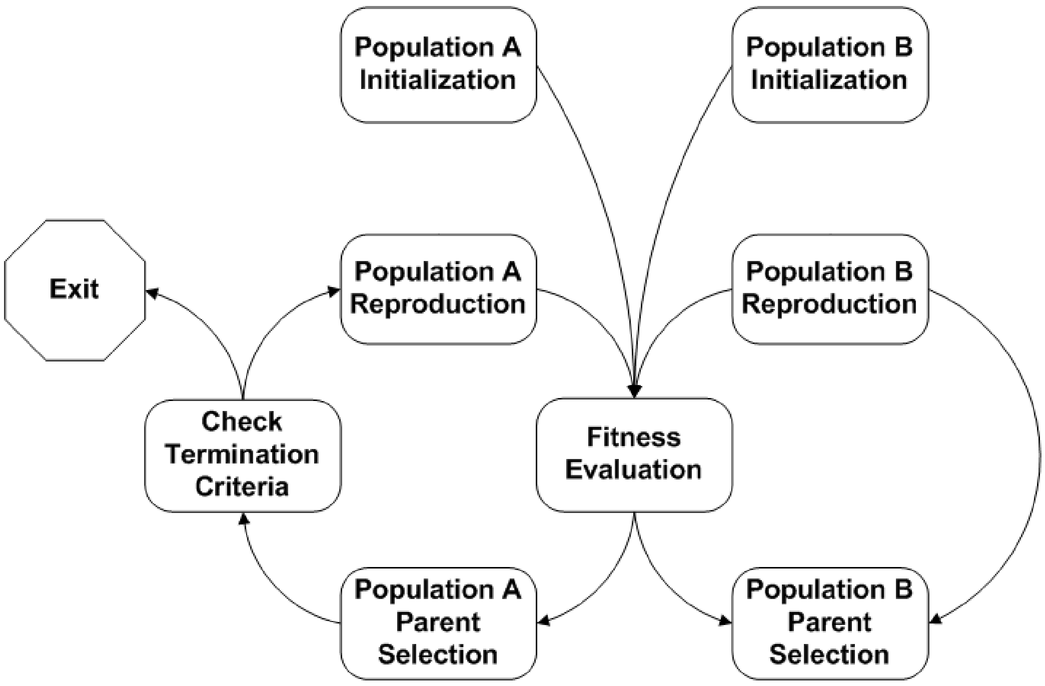

Population A Initialization
Population B Initialization
Exit
Population A Reproduction
Population B Reproduction
Check Termination Criteria
Fitness Evaluation
Population A Parent Selection
Population B Parent Selection

Figure 1.1. Two-population coevolution.

1. over-specialization, where candidate solutions progress on some, but not all, of the underlying objectives [8, 5],
2. disengagement, where one or more evolving populations become too challenging for another population, resulting in a loss of gradient on which to compare individuals [5],
3. inaccurate evaluation, where the fitness of an individual in a particular observed context is not an accurate estimate of its true quality [9] and
4. cycling, where previously seen individuals, determined to be inferior, reemerge in future generations [5, 10].

Many of the problems observed in CoEAs are interrelated (i.e., cycling and over-specialization can both be a side effect of inaccurate evaluation). While different additions to CoEAs have been proposed to address one or more of the studied problems, to the author's knowledge, all such additions have been based on and applied to an underlying evolutionary model.

1.2.1. Interaction Functions. Coevolution replaces the fitness function in traditional evolutionary computation with an interaction function which defines the outcome of interactions amongst the individuals in each population. An individual from each population is supplied to the interaction function which then specifies an outcome of the interaction. Formally, an interaction function, g, is of the form

$$g : P_1 \times \cdots \times P_n \to \mathcal{O}$$

where $P_1 \cdots P_n$ are the n coevolving populations and $\mathcal{O}$ is a set of outcomes. This outcome set may be vector valued, indicating outcomes for each of the n individuals taking place in the interaction and will typically be ordered. Throughout the remainder of this thesis it will be assumed that the set of outcomes are total ordered under the $<$ relation.

In some situations it is convenient to view the interaction function as a measurement table [11]. As an example consider the two-population case. Figure 1.2 shows the measurement table for an interaction function g. The (i, j)-th location in the table represents the outcome of the interaction of individual $i \in P_1$ with individual $j \in P_2$.

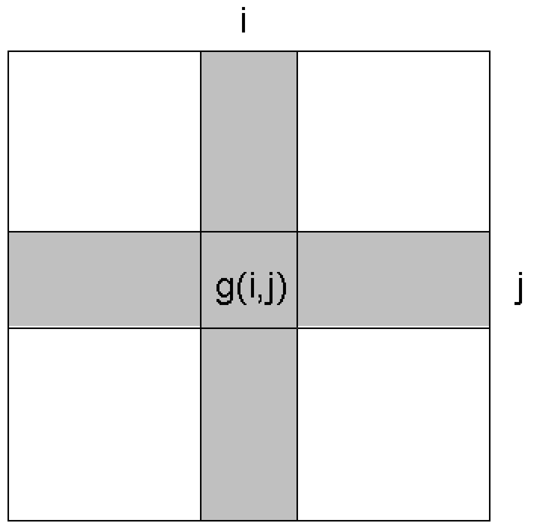


Figure 1.2. Measurement table.

1.2.2. 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 [11]) 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 [12] rather than individuals themselves. 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 [11]. Thus, solution concepts define the problem at hand and serve two purposes: they define the set of solution configurations as well as their form and partition that set into a set of solutions and a set of non-solutions.

This thesis adopts much of the notation from [12] for discussing solution concepts. 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 [11, 13]. The importance 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 [11]. Further it has been shown that solution concepts lacking the property of monotonicity, even if correctly implemented, are expected to exhibit some of the studied pathologies [13].

Several different solution concepts have been used in coevolution literature, and much recent work has focused on designing CoEAs for specific solution concepts [14, 8]. Here a few of them are described in order to illustrate the variety of forms which solution configurations may take.

1.2.2.1. Cooperative coevolution. In many instances an optimization problem can naturally be decomposed into n separate subproblems, or populations, $P_1, \cdots, P_n$, where candidate solutions to each subproblem are evolved simultaneously. In such cases the combination of candidate solutions which maximizes the outcome of the interaction function, g, is desired. Under this solution concept the interaction function can also be viewed as an objective function to optimize. Thus, $C(P_1 + \cdots + P_n) = P_1 \times \cdots \times P_n$.

Formally, under this solution concept the set of solutions is defined by:

$$S = \{C \in P_1 \times \cdots \times P_n : \forall C' \in P_1 \times \cdots \times P_n \;\; g(C) \geq g(C')\}$$

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

1.2.2.2. 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 [4].

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, $g : \mathcal{C} \times \mathcal{T} \rightarrow \mathcal{O}$, 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.

1.2.2.3. Maximization of expected utility. In this solution concept there is also a set of candidate solutions and a set of test cases. 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 its derivative with a nonuniform 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.

1.2.2.4. Nash equilibrium. Game theory provides the concept of the Nash equilibrium solution concept [11]. 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 an n player game, each player i has a corresponding set of behaviors B_i. A Nash equilibrium 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 ΔB_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 equilibrium 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 $\Delta B_1 \times \cdots \times \Delta B_n$.

1.2.2.5. Pareto-optimal set. The Pareto-Optimal Set solution concept borrows ideas from multiobjective optimization. Each test case is treated as a separate objective to be optimized, and the solution set is the non-dominated front.

Formally, a candidate solution C is said to dominate another candidate solution C' if C performs at least as well on all tests as does C' and performs strictly better than C' on at least one test. That is, the relation $dom(C, C')$ is defined to be:

$$\forall T \in \mathcal{T} : g(C,T) \geq g(C',T) \wedge \exists T \in \mathcal{T} : g(C,T) > g(C',T)$$

The solution set is then:

$$S = \{C \in \mathcal{C} : \forall C' \in \mathcal{C} \;\; \neg dom(C', C)\}$$

Thus, the solution set for this solution concept is the entire non-dominated front.

1.2.2.6. 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.

The solution set is given by:

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

Many game theoretic concepts rely on maximizing the worst case outcome of a strategy over all possible combinations of opponents. In such cases the MaxiMin solution concept naturally applies.

1.2.3. Weak Preference Relation. 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 [13]. 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 if T contains only those tests on which it performs optimally. A solution set context, or simply 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 δX.

The weak preference relation is a binary relation on solution configurations [13]. Formally, a candidate solution configuration, α, is weakly preferred to a candidate solution configuration, β, written $\alpha \succ_W \beta$, if every context in which β appears as a solution is a strict subset of a context in which α is a solution.

Definition 1.1 (Weak Preference). *A solution configuration α is weakly preferred to a solution configuration β, written $\alpha \succ_W \beta$, iff for every context X_β with $\beta \in \delta X_\beta$ there is a context X_α 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 [13], and can be viewed as a type of order on the set of configurations[1].

Note that if α and β are two solution configurations returned by CoEAs A and B, respectively, on the same problem instance and $\alpha \succ_W \beta$ then the performance of A can be considered superior to that of B on that particular problem. It is in this manner which the weak preference relation is employed to compare CoEA performance in Section 4.

1.3. CONTRIBUTIONS

This thesis makes three distinct contributions. The primary contribution is a generalization of coevolution to Co-Optimization, presented in Section 2. While there are many parallels between coevolutionary processes and natural evolution, optimization methods other than those based on evolutionary principles may be employed in the interactive fitness setting. Co-optimization allows arbitrary black-box optimization techniques to be employed in interactive domains in place of artificial evolution. This generalization permits matching interactive problem domains to the optimization techniques best suited for them. Section 2 extends the author's earlier work on generalizing coevolution to co-optimization presented in [15].

The second contribution, presented in Section 3, is a methodology based on co-optimization for Critical Infrastructure Protection. A detailed real-world case study on strengthening the electric power transmission system is provided as a demonstration of the methodology. The methodology presented employs co-optimization to

[1]If $\alpha \succ_W \alpha$ was defined for all solution configurations α (the reflexive closure of $\succ$) then the weak preference relation would be a partial order on the set of configurations.

discover system hardenings capable of defending against a wide variety of low-effort, high-impact system faults. The work presented in Section 3 is an extension of the author's previous work presented in [16] and [17].

The No-Free-Lunch (NFL) theorem is a fundamental result in the field of black-box function optimization. In its most basic, and informal, form it states that all search algorithms perform equally well when averaged over all functions to be optimized. The original NFL theorem is applicable only to the class of iterative, data driven search algorithms which optimize an objective function. Due to the interactive nature of fitness evaluation in co-optimization, co-optimization algorithms are not, in general, a member of this class [18]. The third and final contribution of this thesis is a theoretical framework for discussing NFL like results for co-optimization, and thus also for coevolution, presented in Section 4. While it has been shown that, in general, free lunches exist in coevolution [18], and by extension co-optimization, the framework presented in this thesis allows such discussions to be phrased in terms of the type of solution desired. This allows co-optimization algorithms to be naturally classified in terms of the solutions they seek. The NFL framework presented in this thesis will appear in [19].

2. CO-OPTIMIZATION: GENERALIZED COEVOLUTION

Coevolution extends traditional evolutionary computation by measuring an individual's fitness based on interactions with its peers rather than by an external fitness function. The canonical example of coevolution being that of two-population competitive coevolution where one population evolves a set of candidate solutions while the other evolves a set of test cases. Candidate solutions are evolved to maximize their success against the test cases and the test cases are evolved to challenge, and thus promote further advances in, the candidate solutions. This ideally results in an arms race between the competing populations where each population acts as a stepping stone for the other [10].

The basic two-population coevolutionary setup has been shown to be an effective optimization technique [4]. However, the interactive coevolutionary environment has introduced new problems to overcome, such as disengagement, cycling and over-specialization [5]. To address these deficiencies, many additions to the canonical CoEA have been proposed. Diversity maintenance techniques [10] have shown success in overcoming over-specialization [5]. Techniques which incorporate observed behavior from host parasite relationships in biology [5] and other challenge management [20] techniques have been used to combat disengagement. Adding an archive of previously seen individuals [8, 14] has been used to stabilize evaluation and help to prevent inaccurate evaluation and cycling [10]. However, to the author's knowledge, all such additions have been based on, and applied to, an underlying evolutionary model.

While coevolutionary processes have many parallels to natural evolution, techniques other than evolutionary computation may be employed in the interactive fitness setting. Any method of generating new individuals to add to the current populations may be used in place of artificial evolution. As an example consider an algorithm which, for each individual in every population, performs gradient ascent to chose the new individual to replace it.

This thesis presents a generalization of coevolution to *Co-Optimization*, where the use of artificial evolution is replaced with either an arbitrary population based optimization algorithm, or for a population of n individuals, n instances of a non-population based optimization algorithm. The basic principle of interactive fitness is still used in this larger class of algorithms. That is, individuals are still evaluated based on their interactions with their peers. Figure 2.1 diagrams the basic two-population co-optimization algorithm.

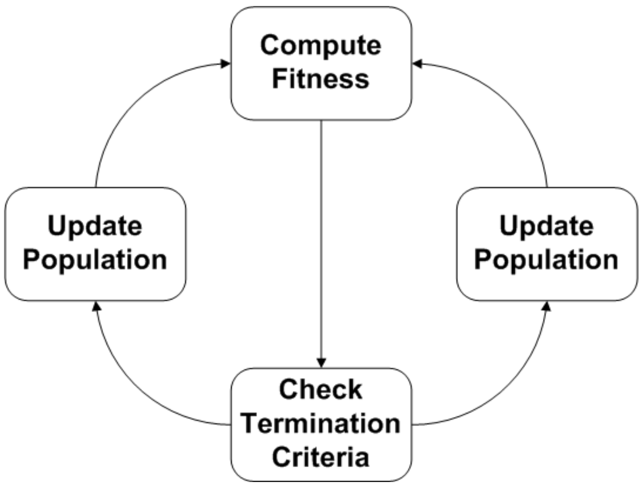


Figure 2.1. Two-population co-optimization.

This generalization allows for matching interactive problem domains to particular optimization algorithms. If a particular domain lends itself to a particular search algorithm, such as simulated annealing, the co-optimization version of simulated annealing could be used in place of evolutionary computation. Also, multiple optimization algorithms could be used in a single instance, where one population is updated based on evolutionary computation, another by simulated annealing, and so on. Thus, if a problem is naturally decomposable into n subproblems, or species, the optimization technique best suited for each individual subproblem can be employed, even in cases of interdependencies between subproblem solutions.

Rather than introducing a new addition to coevolution in order to address a deficiency observed on a particular class of problems, this thesis instead provides a generalized framework within which the most suitable optimization technique for a given problem can be employed.

In general, interactions with multiple individuals are required to generate an accurate estimate of an individual's objective fitness. As such, regardless of the optimization algorithm used, a population, consisting of multiple individuals, is required for each species. Due to this requirement of a population of individuals, the standard terminology from evolutionary computation is employed in the co-optimization setting. Candidate solutions are referred to as individuals in a population and individuals are updated in what are called generations, regardless of the optimization technique employed. Each individual has an associated fitness which is determined by interactions with its peers and co-optimization algorithms are required to attempt to maximize the fitness of their individuals.

An attractive feature of this generalization of coevolution to co-optimization is that the majority of the growing body of theoretical results for coevolution also apply to co-optimization, due to the fact that the majority of such work does not assume an underlying evolutionary model. For example, the notion of solution concepts and the weak preference relation [11, 12], due to the interactive fitness evaluation, naturally apply to the class of co-optimization algorithms.

Also, the numerous additions to the canonical CoEA, such as archives or diversity maintenance techniques, are applicable to arbitrary co-optimization algorithms, as the majority of such techniques do not require an underlying evolutionary model.

2.1. EXAMPLE ALGORITHMS

While the class of co-optimization contains many algorithms, this thesis focuses its attention on the co-optimization versions of gradient ascent, local beam search, simulated annealing and evolutionary computation, in both a canonical and island based form. It should be noted that, in contrast to evolutionary algorithms, both gradient ascent and simulated annealing are non-population based optimization techniques. As such a different set of dynamics is to be expected in the interactive co-optimization environment.

2.1.1. Canonical CoEA.

A basic two-population CoEA is employed as the first test algorithm. This canonical CoEA (C-CoEA) includes no additions to improve performance and is used as a baseline for comparison. Algorithm 1 provides the pseudo-code for the C-CoEA algorithm.

Algorithm 1 C-CoEA

```
function C-CoEA()
 1: for $0 \leq i < numPopulations$ do
 2:    Initialize $population_i$
 3: end for
 4: while not termination condition do
 5:    for $0 \leq i < numPopulations$ do
 6:       Evaluate interactive fitness for $population_i$
 7:       Evolve generation for $population_i$
 8:    end for
 9: end while
10: return $population_i$ $\forall\, 0 \leq i < numPopulations$
```

2.1.2. Co-Gradient Ascent. Co-Gradient Ascent (CoGA) employs a unique instance of gradient ascent for each individual. That is, every generation each individual is updated by gradient ascent. A set of random neighbors is selected for each individual and the member of that set which has the best fitness, in the current context, is selected to replace the current individual in the population. Algorithm 2 provides the pseudo-code for the CoGA optimization technique.

2.1.3. Co-Local Beam Search. In Co-Local Beam Search (CoBEAM) each population employs an instance of the local beam search algorithm. Each generation and for each population, a random neighboring point is generated for each of the k individuals in population and the best k individuals are selected for use as the new population in the next generation. For pseudo-code for CoBEAM refer to Algorithm 3.

2.1.4. Co-Simulated Annealing. Co-Simulated Annealing (CoSA) employs a unique instance of simulated annealing for each individual. Every generation, for each individual, a random neighboring individual is selected and chosen to replace that current individual based upon the cooling schedule. This process is repeated until replacement individuals are selected for all current members of the population. The temperature is held constant throughout the course of a generation and is updated at the beginning of each new generation. Thus, the temperature in each instance of simulated annealing for each individual in the current population, during a given generation, is the same, regardless of the order in which the individuals

Algorithm 2 CoGA

```
function CoGA()
 1: for $0 \le i < numPopulations$ do
 2:     Initialize $population_i$
 3: end for
 4: while not termination condition do
 5:     for $0 \le i < numPopulations$ do
 6:         for each $j \in population_i$ do
 7:             Generate $k$ random mutations of $j$
 8:             Compute fitness for all $k+1$ individuals
 9:             Add the most fit individual to $newPopulation_i$
10:         end for
11:     end for
12:     for $0 \le i < numPopulations$ do
13:         $population_i \leftarrow newPopulation_i$
14:     end for
15: end while
16: return $population_i \ \forall\, 0 \le i < numPopulations$
```

are updated. A linear cooling schedule is used in which the temperature is directly proportional to the number of remaining fitness evaluations. Pseudo-code for the CoSA algorithm is given in Algorithm 4. In Algorithm 4 *COOLING_SCHED()* returns the current temperature as dictated by the employed cooling schedule, and $ACCEPT\_INDIVIDUAL(Temp, k, j)$ determines whether or not individual k should be accepted as a replacement to individual j at temperature $Temp$.

2.1.5. Island Based CoEA. Preliminary results showed that CoSA and CoGA greatly outperformed C-CoEA on the test problems employed (see next section), primarily due to over-specialization of C-CoEA. Lack of diversity is a common problem in both coevolution [10] and traditional evolutionary computation [2] and has been shown capable of causing over-specialization [8]. One method which has been used to promote diversity in traditional evolutionary computing is islanding (see Section 9.3.1 in [2] for an overview of island models). Here a type of CoEA is presented based on the same concept, the island based CoEA (I-CoEA). I-CoEA is identical to the canonical CoEA, except that each population is divided into a number of reproductively isolated islands.

Algorithm 3 CoBEAM

```
function CoBEAM()
1:  for $0 \leq i < numPopulations$ do
2:      Initialize $population_i$
3:  end for
4:  while not termination condition do
5:      for $0 \leq i < numPopulations$ do
6:          $k \leftarrow$ Number of individuals in $population_i$
7:          for each $j \in population_i$ do
8:              Generate random neighbor, $n$, of $j$
9:              Add $n$ to $population_i$
10:         end for
11:         Evaluate interactive fitness for $population_i$
12:         Remove the $k$ least fit individuals from $population_i$
13:     end for
14: end while
15: return $population_i \ \forall\, 0 \leq i < numPopulations$
```

Algorithm 4 CoSA

```
function CoSA()
1:  for $0 \leq i < numPopulations$ do
2:      Initialize $population_i$
3:  end for
4:  while not termination condition do
5:      for $0 \leq i < numPopulations$ do
6:          $Temp \leftarrow COOLING\_SCHED()$
7:          Evaluate interactive fitness for $population_i$
8:          for each $j \in population_i$ do
9:              repeat
10:                 Generate a random neighbor, $k$, of $j$
11:                 Evaluate interactive fitness for individual $k$
12:             until $ACCEPT\_INDIVIDUAL(Temp, k, j)$
13:             Add $k$ to $newPopulation_i$
14:         end for
15:     end for
16:     for $0 \leq i < numPopulations$ do
17:         $population_i \leftarrow newPopulation_i$
18:     end for
19: end while
20: return $population_i \ \forall\, 0 \leq i < numPopulations$
```

While the members of a given island may lose diversity over time, diversity amongst individuals from different islands is preserved. Thus, in general, more islands implies more diversity among the individuals in the population. In the extreme case where the number of islands is equal to the number of individuals, I-CoEA is, for certain parameters, functionally equivalent to CoGA. The pseudo-code for the I-CoEA algorithm is given in Algorithm 5.

Algorithm 5 I-CoEA

```
function I-CoEA()
 1: for $0 \leq i < numPopulations$ do
 2:    for $0 \leq j < numIslands_i$ do
 3:       Initialize $island_j$
 4:    end for
 5: end for
 6: while not termination condition do
 7:    for $0 \leq i < numPopulations$ do
 8:       for $0 \leq j < numIslands_i$ do
 9:          Evaluate interactive fitness for $island_j$
10:          Evolve generation for $island_j$
11:       end for
12:    end for
13: end while
14: return $population_i \ \forall\, 0 \leq i < numPopulations$
```

2.2. NUMBERS GAMES COMPARISON

Here the problem solving capabilities of C-CoEA, I-CoEA, CoGA, CoBEAM and CoSA are compared on several numbers games problems [21, 8, 9].

2.2.1. Problem Definitions.

The numbers game test problems employed here use the maximization of expected utility solution concept.

2.2.1.1. Locally INTransitive.

The Locally INTransitive (LINT) problem [9] is an example of a numbers game, where individuals, both candidate solutions and test cases, are vectors of real numbers, and a candidate solution's fitness is equal to the number of test cases which it solves.

A candidate solution passes all tests which are not greater than or equal to it in all dimensions. However, this is reversed in a region about the candidate solution. Thus, higher values in all dimensions correspond to better global performance; however, search may be driven toward less globally optimal search space regions as locally lower values can lead to increased performance.

Formally, let the candidate solution be the n-dimensional point S. Then S defeats all tests which are less than or equal to $S - \Delta S$ in at least one dimension or which are greater than or equal to S in all dimensions and less than or equal to $S + \Delta S$ in at least one dimension (where $\Delta S = (\Delta s, \cdots, \Delta s)$). Figure 2.2 diagrams the set of tests which a candidate solution S defeats.

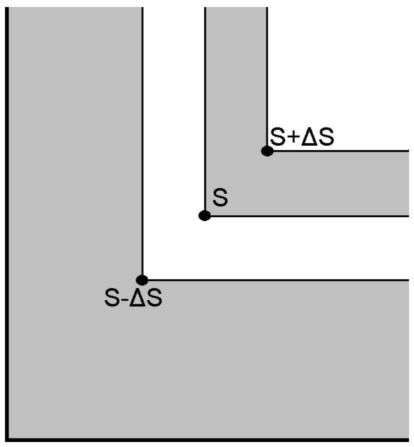


Figure 2.2. The Locally INTransitive (LINT) problem. A candidate solution, S, defeats all test cases in the shaded region.

The variant of LINT employed is the same as was used in [9] where the value of Δs grows as S improves on the underlying dimensions of the problem. Thus, it is relatively easy to make progress when the search begins but becomes increasingly difficult as the search progresses.

For an algorithm to be successful on the LINT problem it must be capable of overcoming the ever growing region of intransitivity. Algorithms attempting to progress on the LINT problem can easily become victims to inaccurate evaluation, as candidate solutions which are closer to the origin may be preferred to those farther away if the current test cases do not correctly discriminate between candidate solutions.

2.2.1.2. Compare-On-One. The Compare-On-One test problem is another example of a numbers game problem. Individuals, both candidate solutions and test cases, are vectors of real numbers. Again a candidate solution's fitness is equal to the number of test cases which it solves. A candidate solution solves a test case if and only if the candidate solution's value on the dimension on which the test case has its largest value is greater than that of the test case. Thus, candidate solutions and tests are compared only based on the test's largest dimension.

Formally, a candidate solution, C, passes a test, T, if

$$C_{T_{MAX}} \geq T_{T_{MAX}}$$

where T_{MAX} is the dimension on which the test T has the largest value.

The Compare-On-One problem has been used in coevolution literature [8, 21] and has been shown to induce over-specialization as tests judge candidate solutions only on a single underlying dimension. Thus, a high level of diversity is necessary among the test cases to adequately evaluate the candidate solutions.

2.2.1.3. Compare-On-All. The Compare-On-All problem is a numbers game identical the Compare-On-One problem, except that individuals are compared on all underlying dimensions instead of just a single dimension [21].

Formally, a candidate solution, C, passes a test, T, if

$$\forall i \in \{1, 2, \cdots, n\} \;\; C_i \geq T_i$$

where the individuals are n dimensional vectors of real numbers.

The Compare-On-All problem is, in a sense, a simpler version of both the Compare-On-One and LINT problems as it shares the underlying goal of both other problems, without the added obstacles. As such, algorithms attempting to solve the Compare-On-All problem are not as susceptible to the problems of inaccurate evaluation and over specialization as they are with the Compare-On-One and LINT problems. However, disengagement is still a possible concern.

2.2.2. Results. All results are statistically significant with an α of 0.05 on a two-tailed t-test assuming unequal variances, unless otherwise stated, and all graphs depict the performance of each algorithm averaged over all runs. Table 2.1 shows the parameter values used. On each problem I-CoEA is used with island sizes of 5, 8, 10 and 20; however, only the island size which performed best is included in the plots for comparison with the other algorithms.

Table 2.1. Parameters for C-CoEA, I-CoEA, CoSA, CoGA and CoBEAM used in the LINT, Compare-On-One and Compare-On-All problems.

Parameter	Value
Populations	2
Population size	40
CoEA recombination	One point crossover
Mutation	Addition of a uniformly selected random number in the interval $[-3, 3]$ to one allele
CoGA number of evaluated neighboring individuals	1
CoEA offspring	40
CoEA parent and survival selection	Binary tournament selection
Fitness evaluations	6 million
Number of runs	60
Population initialization	Each allele is a uniform random number from the interval $[0, 10]$
Dimensions	2
Minimum value in any dimension	0.0
Maximum value in any dimension	1000.0

For each of the three test problems, individuals which have high values in all dimensions are more fit, in a global sense, than those which have lower values. As was done in [8], the least value in all dimensions is used as a measure of objective fitness for all test problems. In cases of over-specialization where, for example, a particular algorithm finds candidate solutions which have very large values on one dimension but not the others, that algorithm will have a low overall objective performance due to the neglect of the other underlying problem dimensions.

2.2.2.1. Locally INTransitive. For the LINT problem, I-CoEA with eight reproductively isolated islands outperformed all other algorithms. CoSA and CoGA produced the second highest average values, with no statistically significant difference between the two. C-CoEA and CoBEAM performed the worst, with no statistically significant difference between the two.

While I-CoEA experienced an increase in performance by increasing the number of reproductively isolated islands from five to eight, the objective performance decreased when the number of islands was increased to ten, indicating that, for this problem, a balance between diversity (i.e., the number of islands) and a sufficient number of individuals per island for evolution is needed for optimal performance.

Figure 2.3 shows a plot of the objective fitness value for the most fit individual for each search algorithm averaged over all runs.

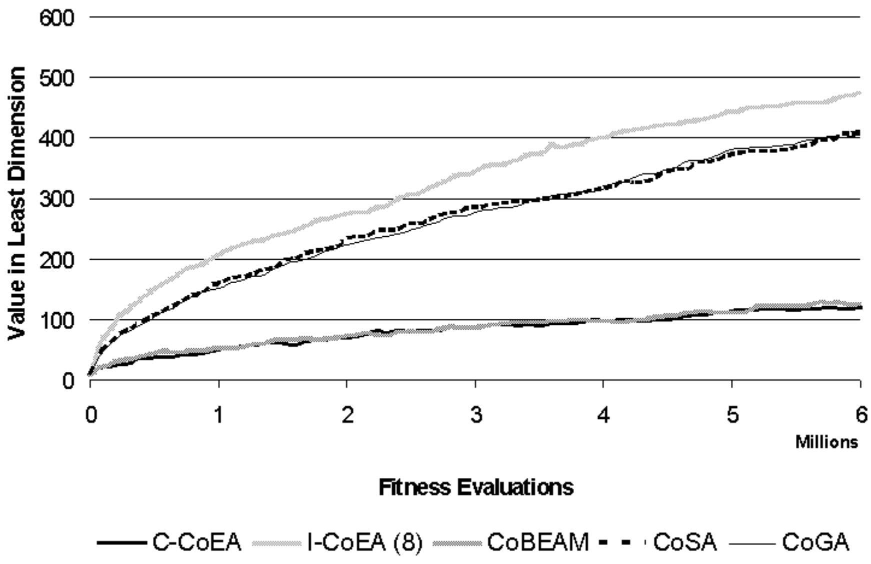


Figure 2.3. Performance of C-CoEA, I-CoEA, CoSA, CoGA and CoBEAM on the Locally INTransitive problem.

2.2.2.2. Compare-On-One. For the Compare-On-One problem, CoGA performed better than all other algorithms tested, followed by I-CoEA with twenty reproductively isolated islands. CoSA performed third best followed by C-CoEA and CoBEAM, for which there was no statistically significant difference in performance.

For this problem, each increase in the number of islands in the CoEA was accompanied by an increase in objective performance. Figure 2.4 shows a plot of the objective fitness value of the most fit individual for each search algorithm averaged over all runs.

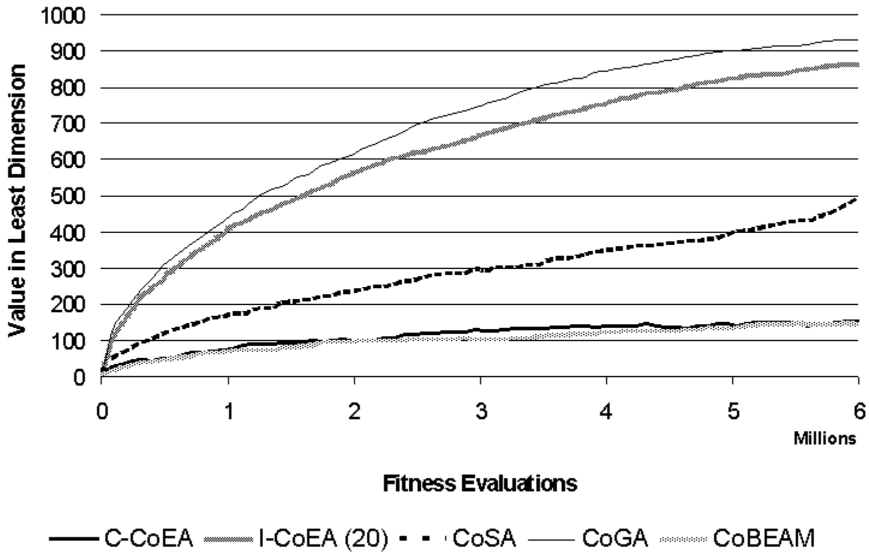


Figure 2.4. Performance of C-CoEA, I-CoEA, CoSA, CoGA and CoBEAM on the Compare-On-One problem.

2.2.2.3. Compare-On-All. The relative performance of each algorithm on the Compare-On-All problem was identical to that of the Compare-On-One problem: CoGA performed the best, followed by I-CoEA with twenty reproductively isolated islands followed by CoSA and finally by both C-CoEA and CoBEAM, for which there was no statistically significant difference in performance. Figure 2.5 shows a plot of the objective fitness of the fittest individual for each search algorithm averaged over all runs.

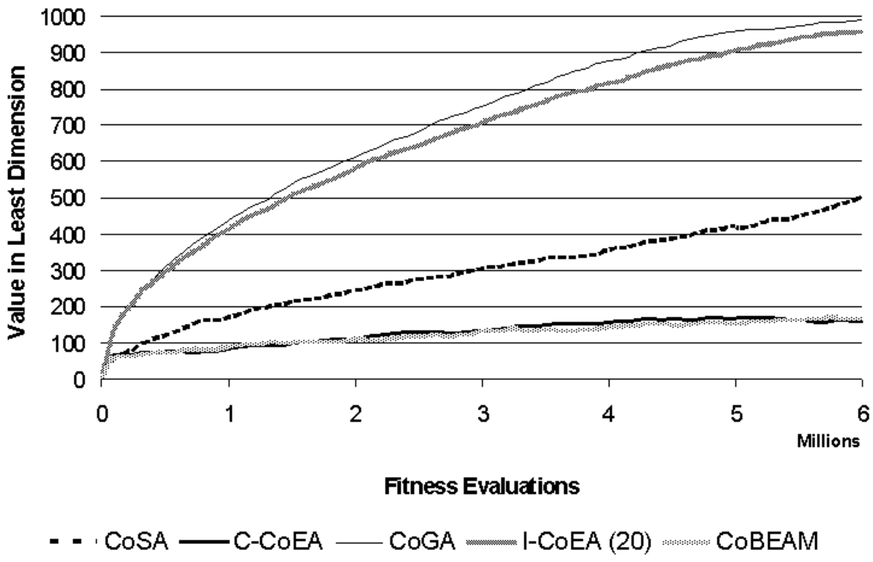


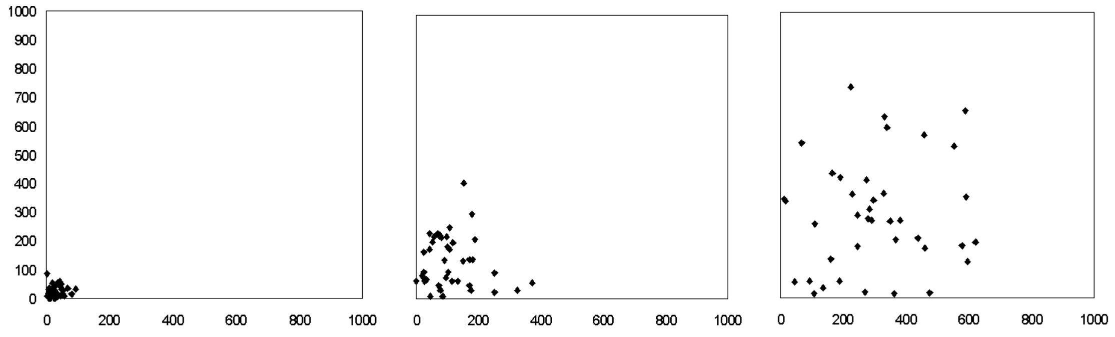
Figure 2.5. Performance of C-CoEA, I-CoEA, CoSA, CoGA and CoBEAM on the Compare-On-All problem.

2.3. POPULATION DIVERSITY

Maintaining adequate levels of diversity amongst individuals is essential in co-optimization and coevolution to allow for accurate evaluation. Many additions to canonical coevolution have been suggested to promote diversity, see [5] for an summary of such techniques.

Here the level of diversity among individuals in each co-optimization algorithm is compared. As individuals in numbers game problems are vectors of real numbers it is easy to visualize the diversity in a population by a plotting its members. The diversity of each algorithm's individuals is plotted over the course of a run on the Compare-On-One problem as it was designed to induce over-specialization, and thus a loss in diversity is expected as individuals over-specialize and neglect one or more underlying problem dimensions. For each algorithm three plots are given which demonstrate how the individuals progress over time on one run of the Compare-On-One problem. It should be noted that the three plots do not correspond to the same number of fitness evaluations or the same number of generations for all algorithms. They are simply representative of the way in which each algorithm explores the search space. Refer to Figure 2.4 for a comparison of average algorithm performance in terms of fitness evaluations.

Figure 2.6 plots the individuals from a CoSA population early in the run, during the middle of the run and at the end of the run. CoSA maintains a fairly uniformly distributed population of individuals in the visited area of the search space. Such a distribution of individuals allows accurate evaluation as the population progresses equally on all dimensions. Further there are individuals which progress equally on all dimensions, as desired.



(a) Early in Run (b) Middle of Run (c) End of Run

Figure 2.6. CoSA diversity. Each subfigure plots the individuals from a CoSA population over the course of a single run.

Figure 2.7 shows a plot of a population from a run of CoGA, where a single neighboring individual for each current member of the population, is generated each generation. CoGA displays similar behavior to CoSA in that members of the population are fairly uniformly distributed throughout the visited area of the search space.

The number of neighboring points generated each generation for every individual has an interesting affect on the behavior of CoGA. Figure 2.8 plots the individuals from a CoGA population, where ten neighboring individuals are generated each generation (CoGA (10)), early in the run, during the middle of the run and at the end of the run. As can be seen in Figure 2.8(b), the CoGA population, as a whole, progresses on all underlying dimensions; however, the individuals within the population progress

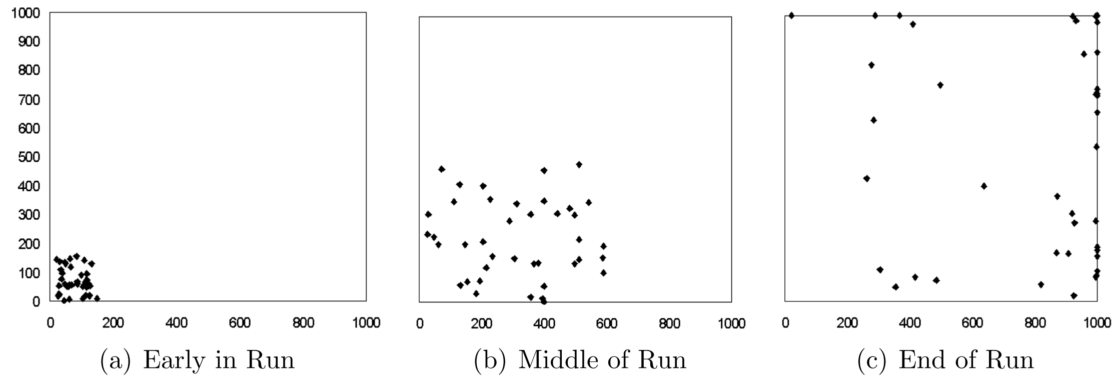


Figure 2.7. CoGA diversity. Each subfigure plots the individuals from a CoGA population over the course of a single run.

predominantly on a single dimension, resulting in a lower objective fitness. This is most likely due to the fact that there is a greater chance for random movement when a single neighboring individual is generated than when many neighboring individuals are. To illustrate this consider the two dimensional case. For a given individual, it may be easier to generate a more fit individual by a move along dimension 1 than by dimension 2, for example increasing fitness by moving along dimension 2 might require a large jump. If only a single individual is generated then it is equally likely that it will have been created by a move along either dimension. Half of the time it will be created by a move along dimension 2, which in the current context would be considered equally fit to its parent individual, even though it may have increased or decreased in objective fitness. As both individuals have equal fitness both are equally likely to surive to the next generation, resulting in occasional random movements which do not increase or decrease fitness. When a large number of neighboring individuals are generated it is unlikely that an increase along dimension 1 will not be created, resulting in less random movement and, in this case, more over-specialization.

The manner in which CoGA (10) explored the search space would provide accurate evaluation as it progressed on all of the underlying objectives, or dimensions, in the sense used in [22].

The plots of the individuals from both C-CoEA, Figure 2.9, and CoBEAM, Figure 2.10, show the same lack of diversity. All members of the population are tightly clustered and progress along a single dimension. This behavior results in both

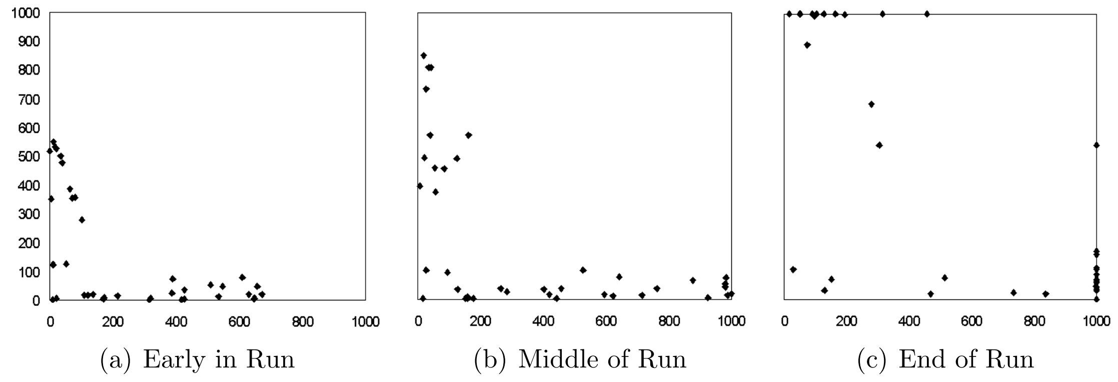


Figure 2.8. CoGA (10) diversity. Each subfigure plots the individuals from a CoGA (10) population over the course of a single run.

low objective fitness as both algorithms quickly over-specialize on a single dimension as well as inaccurate evaluation as this over-specialization promotes advances along a single dimension.

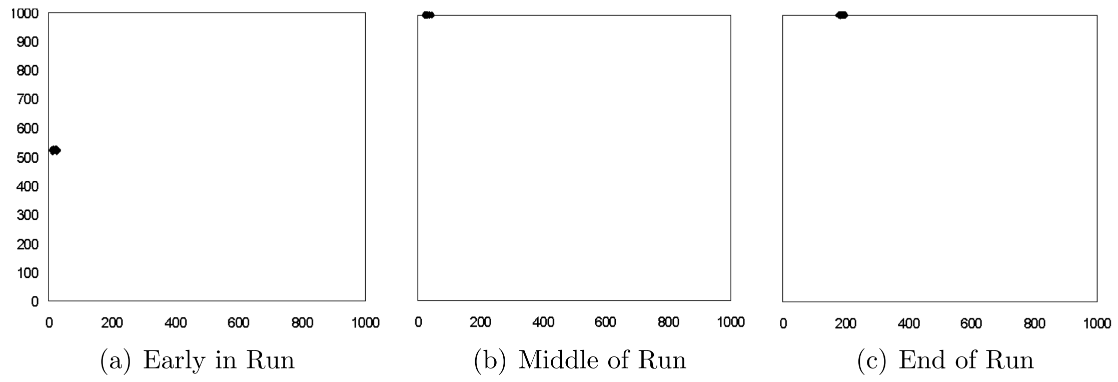


Figure 2.9. C-CoEA diversity. Each subfigure plots the individuals from a C-CoEA population over the course of a single run.

Figure 2.11 plots the individuals from a I-CoEA population with ten reproductively isolated islands early in the run, during the middle of the run, and at the end of the run. While the individuals in each island are tightly coupled together, the islands themselves progress to different areas of the search space.

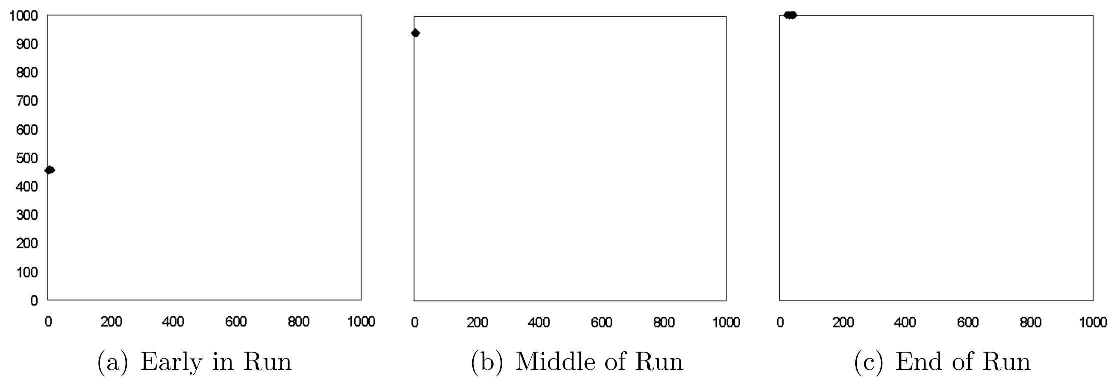

(a) Early in Run (b) Middle of Run (c) End of Run

Figure 2.10. CoBEAM diversity. Each subfigure plots the individuals from a CoBEAM population over the course of a single run.

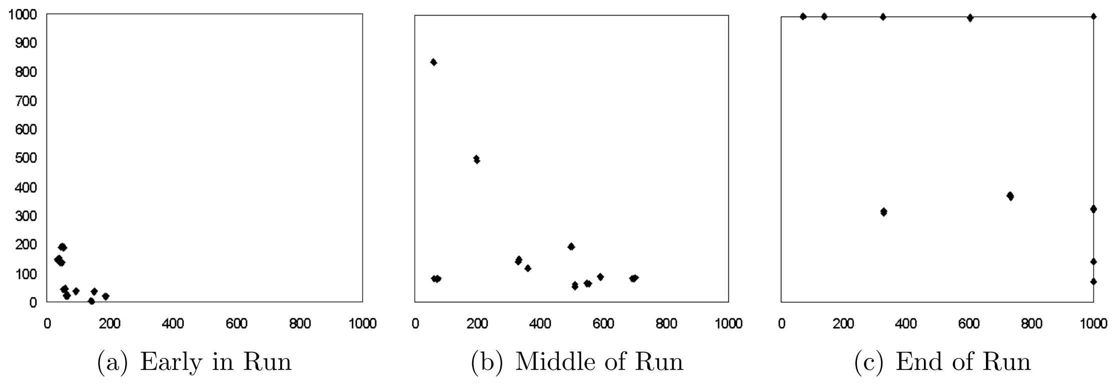

(a) Early in Run (b) Middle of Run (c) End of Run

Figure 2.11. I-CoEA (10) diversity. Each subfigure plots the individuals from a I-CoEA (10) population over the course of a single run.

2.4. EFFECTS OF DISENGAGEMENT

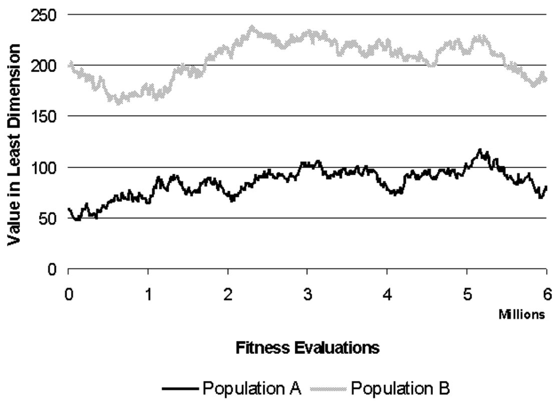
Disengagement is a well studied problem observed in coevolution, which results when one or more populations become too challenging for another resulting in a loss of gradient on which to compare individuals [5]. Without such a gradient, effective search is impossible and degrades into random drift. Here the effects of disengagement are investigated on C-CoEA, I-CoEA, CoSA, CoGA and CoBEAM. Each co-optimization algorithm is used on the Compare-On-All problem where population A is initialized such that each dimension of each individual is a uniform randomly selected number between 50 and 60 and in population B each dimension of each individual is a uniform randomly selected number between 190 and 200. This insures that both populations start off in a highly disengaged state, with all members of population B defeating all members of population A. The results presented here only demonstrate how certain co-optimization algorithms behave in a disengaged state and not how likely each algorithm is to reach such a state.

Figures 2.12(a), 2.12(b), 2.12(c), 2.12(d) and 2.12(e) show the best objective fitness values of each population when started off in a disengaged state for C-CoEA, I-CoEA, CoSA, CoGA and CoBEAM, respectively.

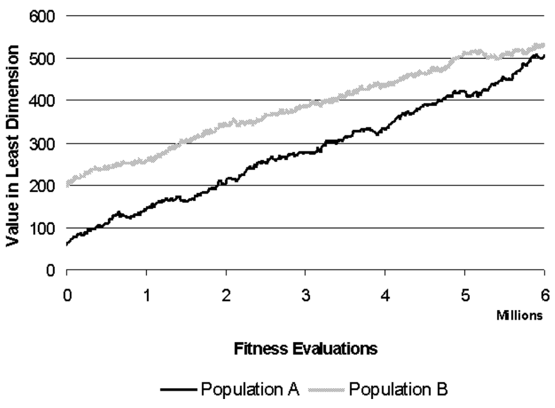
I-CoEA, CoSA and CoGA were still able to make significant progress while both C-CoEA and CoBEAM were caught in a period of random drift for the entirety of most runs. As one would expect, the progress of I-CoEA and CoGA was significantly hindered by starting it out in a disengaged state, compared to starting in an engaged state (refer to Figure 2.5). However, the progress of CoSA when started out in a disengaged state was nearly identical to that of CoSA started out normally.

CoSA and CoGA were able to make progress when started from a disengaged state due to the fact that they are non-population based optimization techniques, as opposed to C-CoEA and CoBEAM which performed poorly when disengaged. Even if only a single individual, in a non-population based technique, from population A is able to become competitive with the members of population B, in a setting of disengagement, the search is able to progress with that individual alone.

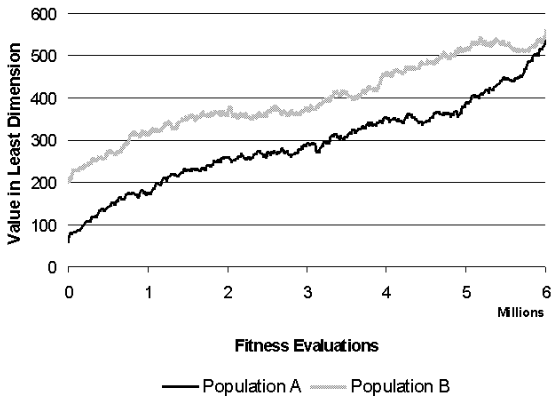
The progress made by I-CoEA was dependent upon the number of islands used. Figure 2.13 shows plots of I-CoEA's performance for 5, 8, 10 and 20 islands. As the number of islands increased, and thus as I-CoEA became less population based, the objective performance improved.



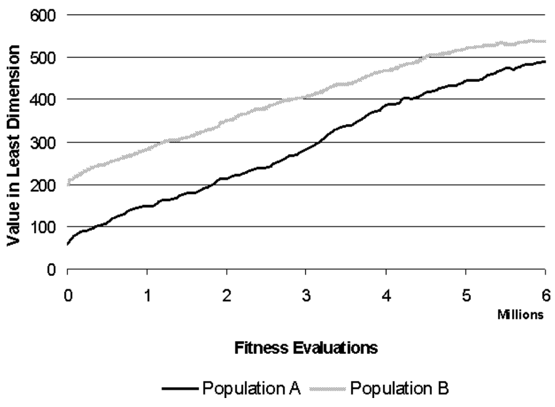
(a) C-CoEA



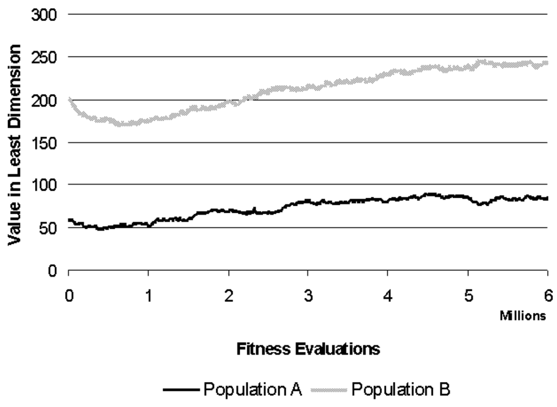
(b) I-CoEA(20)



(c) CoSA



(d) CoGA



(e) CoBEAM

Figure 2.12. Effects of disengagement on the Compare-On-All Problem.

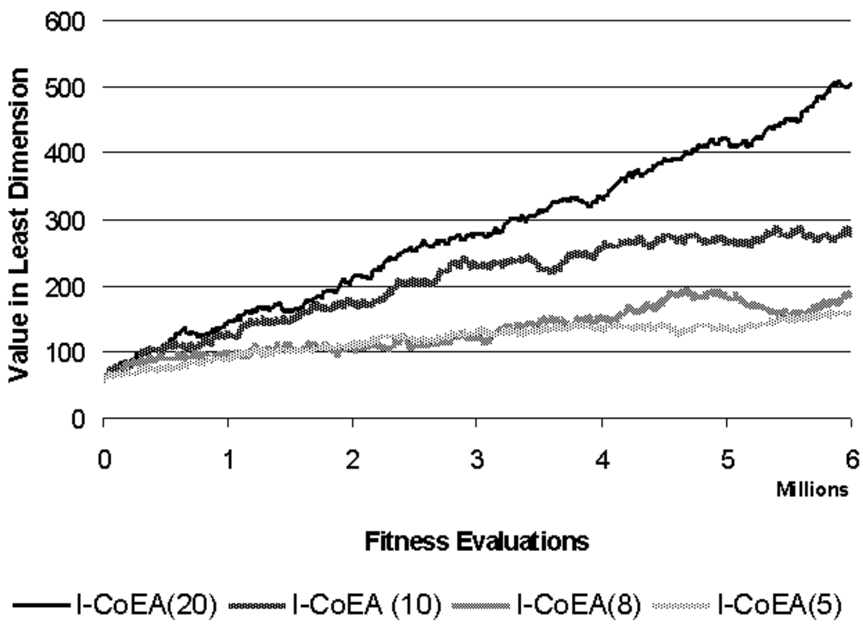


Figure 2.13. Effect of I-CoEA island size on disengagement.

2.5. ITERATED PRISONER'S DILEMMA

The Prisoner's Dilemma is a classic example of an interactive problem domain [23]. The Prisoner's Dilemma is formulated as follows: Two individuals Alice and Bob are being interrogated separately by the police for a crime they committed together. No communication between the two is allowed. Both Alice and Bob have two options. They can either defect and accuse their accomplice of committing the crime, or they can cooperate with their accomplice and remain silent. If both cooperate then both will receive 2 years in prison, if both defect then both will receive 4 years. If one individual defects and the other cooperates, then the defecting individual will be allowed to leave free, while the cooperating individual will receive 5 years in prison.

Abstractly, the Prisoners Dilemma can be described by the payoff matrix in Table 2.2, where each player attempts to maximize the number of points they get, where the number of points is 5 minus the number of years in prison.

In the Iterative Prisoner's Dilemma (IPD) a number of iterations of the basic prisoner's dilemma is played, and a memory of past iterations is maintained so that

Table 2.2. Prisoner's Dilemma payoff matrix.

		Player B	
		Cooperate	Defect
Player A	Cooperate	3,3	0,5
	Defect	5,0	1,1

strategies can use information from past games to decide whether to cooperate or to defect. In the experiments presented here ten iterations are played and, as done in previous IPD work [23], each strategy has a memory of the past three iterations.

Since each strategy retains a memory of the past three games, there are 64 possible three game sequences for which the strategies must be capable of determining an action. Individuals are represented by bit strings of 70 bits. Where the *i*-th bit represents the action which the strategy takes when the *i*-th sequence of three games is observed: either to defect or to cooperate. The final six bits are used to represent a hypothetical previous sequence of three games used by the strategy to determine to seed the initial three game memory for determining actions prior to the completion of the first three actual games. The parameters for each search algorithm are given in Table 2.3

Table 2.3. Parameters for each algorithm used in the Iterated Prisoner's Dilemma.

Parameter	Value
Populations	2
Population size	40
CoEA recombination	One point crossover
Mutation	Bit flip of a single bit
CoGA # of neighbors	1
CoEA offspring	40
CoEA parent and survival selection	Binary tournament selection
Fitness evaluations	1 million
Population initialization	Each bit 0 or 1 with 50% probability

The amount of emergent cooperation between individuals over time is measured. Under each algorithm, individuals converged to cooperative states, with all but CoBEAM finding populations which cooperated in more than 80% of the games. Figure 2.14 shows a plot of the percentage of games in which the first player chose to cooperate. While the C-CoEA quickly converged to cooperative solutions, CoSA and CoGA eventually found more cooperative solutions, with CoGA finding the most cooperative set of individuals on average.

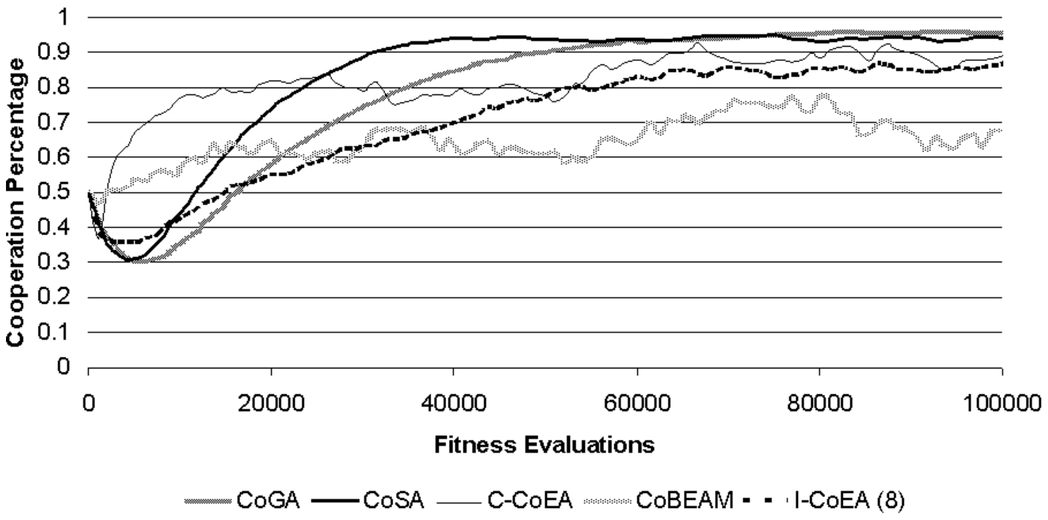


Figure 2.14. Amount of emergent cooperation of C-CoEA, I-CoEA(8), CoSA, CoGA and CoBEAM on the Iterative Prisoner's Dilemma.

As observed in previous studies [23] all algorithms exhibited the same initial tendency towards defecting strategies; however, this initial tendency towards defective strategies was muted in both CoBEAM and C-CoEA, compared to the other algorithms, ith both algorithms quickly producing strategies which cooperated more than fifty percent of the time.

2.6. DISCUSSION

This section has introduced the novel class of co-optimization algorithms. Co-optimization extends coevolution to arbitrary black-box function optimization techniques. Three examples of non-evolutionary co-optimization algorithms are presented:

CoGA, CoBEAM, and CoSA. The performance of all three are compared to that of C-CoEA and I-CoEA on three numbers game problems as well as on the Iterative Prisoner's Dilemma. It is found that CoSA and CoGA have better performance on the three numbers game test problems than does C-CoEA and do not suffer from the problems of over-specialization and inaccurate evaluation that C-CoEA does on the numbers game test problems. The performance of CoBEAM was nearly identical to that of C-CoEA on numbers game problems, and the performance of I-CoEA was dependent upon the number of islands used. On the Compare-On-One and Compare-On-All problems greater numbers of islands always corresponded to better performance; however, on the LINT problem a trade-off was observed between the number of islands and adequate number of individuals per island for evolution.

The effects of disengagement are investigated on the different co-optimization algorithms using the Compare-On-All problem. CoSA, CoGA and I-CoEA were all able to effectively reengage and make progress, with greater numbers of islands in I-CoEA corresponding to better progress; however, C-CoEA and CoBEAM remained disengaged throughout the entirety of most runs. This is hypothesized to be a result of non-population based optimization techniques naturally maintaining greater amount of diversity between the individuals in the population.

Finally, the different co-optimization algorithms are tested on the Iterative Prisoner's Dilemma problem. While all algorithms produced strategies which were cooperative in games amongst themselves, they varied in terms of the level of cooperativeness as well as how quickly such strategies were generated. All algorithms but CoBEAM produced a high level of cooperativeness.

The generalization of coevolution to co-optimization allows interactive problem domains to be matched with the optimization technique best suited for them. If deficiencies are observed in a particular optimization techniques on a given problem then that technique may be swapped out for a different method which does not exhibit the observed deficencies. An example of when this ability might be useful is in the case of disengagement. In the experiments detailed in this section, the non-population based co-optimization algorithms, CoSA and CoGA, were able to perform well in the face of complete disengagement, while the population based algorithms remained in a period of random drift for the entirety of most runs.

3. CRITICAL INFRASTRUCTURE PROTECTION

The world is increasingly dependent on critical infrastructures such as the electric power grid, water, gas and oil transport systems. Due to economic, social, and environmental constraints, the current rate of infrastructure expansion is inadequate to continue meeting the increasing demand [24]. This trend leaves these systems less resilient to external faults, both accidental and malicious, than ever before. As a result of this increased vulnerability, many critical infrastructures are becoming susceptible to cascading failures, where an initial fault caused by an external force may induce a domino-effect of further component failures. A well known example of such a cascading failure is the 2003 Northeast Blackout during which a large portion of the Northeastern United States and Ontario, Canada lost power [25, 26]. Because of the increasingly interconnected nature of infrastructures, cascading failures now threaten to jump infrastructure borders causing further damage.

These trends combined raise the spector of a well targeted attack bringing down an entire infrastructure or system of interconnected infrastructures, resulting in a devastating economic blow and potentially a significant loss of life. Traditional infrastructure risk analysis methods, often relying on Monte Carlo sampling of fault scenarios, are not sufficient against intelligent adversaries. Instead, systematic analysis based on worst-case fault scenarios is essential.

The incorporation of intelligent control devices in infrastructures provides the ability to dynamically balance system use during normal operating conditions and redistribute system resources during fault scenarios. Both alone and coupled with topological changes, these devices offer the possibility of increasing system resilience to cascading failures. However, strategic placement of such devices and intelligently chosen topological changes are essential to fully reap the benefits they offer.

The problems of finding optimally balanced hardenings and worst-case disaster scenarios are interdependent and their solution spaces share the characteristics of:

1. combinatorial complexity, making exhaustive search on even moderate size systems infeasible, and

2. non-linear dependencies between solution components, resulting in many local optima which defeat most traditional search algorithms.

Correct functioning of the electric power transmission system is a prerequisite for the correct functioning of most other critical infrastructures. Such Level 1 infrastructures [27] require significant attention due to the potential disruption other infrastructures face if they are disabled.

This section presents a co-optimization methodology for Critical Infrastructure Protection (CIP) as well as a demonstration of this methodology on an electric power transmission system real-world case study.

3.1. METHODOLOGY

A hardening is a set of modifications designed to make a system more resilient to faults. System hardenings should effectively minimize damage caused by a wide range of faults, including worst-case scenario faults, and the hardening which provides the most economic protection for the least economic overhead should be adopted. To measure the economic protection of a hardening, the hardening must be evaluated in the face of system faults. More formally, if a fault f is expected to occur E_f times throughout the lifetime of the system and inflicts $L(h, f)$ dollars in damage with the hardening h in place and $L(\emptyset, f)$ dollars in damage without the hardening in place, then the hardening h saved $E_f \times (L(\emptyset, f) - L(h, f))$ dollars over the lifetime of the system. The most desirable hardenings are those capable of maximizing that difference over large numbers of likely faults.

Equation (1) provides a measure of the total cost of a hardening over the lifetime of the system. Minimizing Equation (1) is equivalent to maximizing $E_f \times (L(\emptyset, f) - L(h, f))$ over all faults, as $E_f \times L(\emptyset, f)$ is independent of the hardening used.

$$F(h) = C(h) + \sum_{f \in \Omega} E_f \cdot L(h, f) \tag{1}$$

where $F(h)$ measures the expected value of the total amount of economic cost the system will require throughout the lifetime of its use with hardening h in place, $C(h)$ is the overhead cost of the hardening, such as installation, Ω is the set of all possible faults the system might incur, E_f is the expected number of times fault scenario f will occur throughout the lifetime of the system, and $L(h, f)$ is the expected monetary damage caused to the system by fault scenario f with hardening h in place. The minimal values of $F(h)$ correspond to the hardenings with the best cost-benefit ratio. One means by which to obtain values of E_f is to use Model-Based Vulnerability Analysis presented in [27].

Equation (2) shows the analogous equation for evaluating a fault over all possible hardenings.

$$G(f) = \sum_{h \in \Sigma} [E_f \cdot L(h, f) + C(h)] \tag{2}$$

where $G(f)$ measures the expected monetary damage caused by fault scenario f averaged over all possible hardenings from the set Σ. Thus, the maximal values of $G(f)$ represent those faults which, on average, cause the most damage to the system with an arbitrary hardening in place. When evaluating the effects of a fault in a real-world situation, a hardening is already in place and the fault need only be evaluated against that specific system configuration and not against all possible hardenings. However, in this methodology faults are being optimized to promote advances in the hardenings. Faults which provide a challenge for all of the current hardenings are desired. As such faults need to be evaluated against all of the current hardenings.

Equations (1) and (2) represent a variation of the maximization of expected utility solution concept where a non-uniform distribution over the test cases is employed.

Ideally, each hardening would be evaluated against the entire range of faults and a hardening with the least expected cost, $F(h)$, selected; likewise the worst scenarios or faults, f, can be determined from Equation (2) by selecting those faults with the greatest expected cost, $G(f)$. In practice it is not feasible to evaluate a hardening against all possible faults, nor a fault against all possible hardenings.

The CIP methodology presented in this thesis employs co-optimization to simultaneously optimize system hardenings and faults. The methodology presented here employs unique populations for both system hardenings and faults. In such a manner the system hardenings are challenged by increasingly damaging faults, resulting in, ideally, hardenings which withstand a wide range of faults, and cover general areas of the system in need of strengthening. By each population using its opposing population as stepping stones [10], the protective ability of hardenings and damaging effects of faults are forced to increase incrementally.

3.2. RELATED POWER SYSTEM WORK

Much work has gone into hardening the electric power transmission system [28, 29]. A promising approach to hardening power transmission systems is through the use of power system control devices, in particular the family of power electronics-based controllers known as Flexible AC Transmission System (FACTS) devices [30]. One of the most powerful types of FACTS devices is the Unified Power Flow Controller (UPFC). The UPFC is attached between a bus and a transmission line and can be used to control the phase angle, bus voltage, and line reactance. With intelligent placement and control, UPFCs have been shown capable of preventing cascading failures [31].

Previous work has gone into both optimizing the placement of UPFCs in an electric power system and determining UPFC control parameters during operation. Sequential Quadratic Programming (SQP) has previously been employed to determine long term UPFC set points [32, 33]. SQP is a hill climbing method to solve constrained non-linear optimization problems. In SQP a series of quadratic programming subproblems are formed, whose solutions converge to the solution of the original problem. SQP has been employed to find UPFC set points which minimize a performance index (PI) [32]:

$$PI = \sum_{i \in lines} w_i \cdot \left(\frac{S_i}{S_i^{max}} \right)^n \tag{3}$$

where S_i is the current power flow through line i, S_i^{max} is the maximum rated capacity of line i, and n is a free parameter no less than 1. The ratio $\frac{S_i}{S_i^{max}}$ is raised to the power n to more heavily punish heavily overloaded lines. By increasing n, more importance is given to those lines which are heavily overloaded. Weights w_i provide a means to assign varying levels of importance to different lines, for example it may be desirable to minimize the overload on larger lines, as their removal has more potential to affect large portions of the system, before the overload on smaller lines.

In general, smaller PI values indicate a more balanced use of the system transport capacity, as skewed flow distributions are punished due to the larger percent capacity usage on some lines. SQP is used to determine UPFC set points such that the PI metric is minimized. It should be noted that SQP is guaranteed to find optimal set points if the PI metric is convex; however, it is currently not known for which operating conditions the PI metric is a convex function and thus the set points found by SQP may only be local, and not global, minima [32].

Previous work has proposed placing UPFCs in an electric power transmission system to minimize a quadratic cost function consisting of generation cost and UPFC investment cost [28]. An algorithm was given to determine the optimal UPFC placement to minimize the cost function and a case study is presented on the IEEE 14 bus system. While the optimal UPFC placement was successfully found for this system, the cost function used considers only normal operating conditions and does not take into consideration external faults.

Evolutionary computation has been previously applied to the problem of UPFC placement in a power transmission system [34, 33, 1]. UPFC placements have been evolved to minimize the number of line overloads over all single line faults [34] without taking into consideration the additional failures induced by the original faults. UPFC placements have also previously been evolved to minimize the sum of the PI metric, Equation (3), over all single line faults [33], and the results were compared against a greedy heuristic approach in which the best fifty placements of a single UPFC are combined to form $\binom{50}{2}$ placements of two UPFCs, $\binom{50}{3}$ placements of three UPFCs, and so on. The evolved placements of k UPFCs were compared against the best heuristic placement out of the $\binom{50}{k}$ such placements and found to be better in terms of the PI metric, Equation (3).

3.3. POWER SYSTEM MODEL

A steady state model of a subset of the IEEE 118 bus test system[2] is used as the testbed for all experiments presented here. The Newton-Raphson iteration method with optimal multiplier [35] is employed to solve the polar form of the steady state system of loadflow equations, further referred to as loadflow, to determine the power

[2] `http://www.ee.washington.edu/research/pstca/pf118/pg_tca118bus.htm`

flow through each line in the system. An iterative load shedding method is used to restore solvability in unsolvable situations such as when demand exceeds system capacity [35]. If after ten iterations of the load shedding algorithm a solution is not found, it is assumed a complete blackout has occurred. System islanding is handled by employing multiple instances of steady state loadflow. Once islanding has occurred, the line flows are solved for each island by a separate instance of loadflow.

To more accurately simulate some of the slower acting dynamics of an electric power transmission system, specifically cascading failures, several iterations of the steady state loadflow are performed to account for line overloads and additional contingencies induced by the initial faults.

Each line in the system has a maximum power capacity rating, specifying the amount of power it is capable of carrying. Whenever line i's current power flow, S_i, is greater than 100% of its rated capacity, S_i^{max}, its remaining amperage, A_i, is decreased by the amount by which its power flow is greater than 100% of its rated capacity, $(S_i - S_i^{max})$, times the time spent at that flow, Δt. When line i's current power flow is less than its rated capacity, its remaining amperage is increased by a cooling factor up to the maximum, A_{max}. Line i fails when $A_i = 0$. Since in general when steady state loadflow converges, the bus voltages are close to their nominal values, the direct comparison between transmission line power flows in watts and line current ratings in amps is not a significant source of error.

In a real-world power system, the total amount of power delivered over the period of time from t_0 to t can be determined by taking the integral over that time period of the power delivered at each instance. In the employed discretized steady state model this corresponds to a sum over each time step of the power delivered during that time step.

In order to test the proposed methodology, values for the various system parameters were chosen close enough to reality to allow validation of our methodology. A_{max} was chosen such that a line running at 200% of its rated capacity fails in roughly 30 seconds and the cooling factor was choosen to be 50. With these values a line on the brink of failure recovers fully in just under two minutes. The change in time, Δt, is taken to be one second. With the selected system parameters, a finer time scale would only increase the computational complexity without adding to the realism of the simulation. The total amount of power delivered over the course of a 12 hour period is simulated. Algorithm 6 details the power system simulation used.

Algorithm 6 Power System Simulation

```
function LoadServed(h,f)
1: Apply hardening $h$ to the system
2: Apply fault $f$ to the system
3: $max\_time\_step = 60 \cdot 60 \cdot 12$
4: $A_{max} = 30 \cdot 60$
5: $cool\_factor = 50$
6: for each line $i$ do
7:     $A_i = A_{max}$
8: end for
9: Calculate steady state loadflow
10: $time\_step = 0$
11: while $time\_step < max\_time\_step$ do
12:     Increment $time\_step$
13:     if $A_i \leq 0$ then
14:         Decommission line $i$
15:     end if
16:     if any line decommissioned this $time\_step$ then
17:         Recalculate steady state loadflow
18:     end if
19:     for each line $i$ do
20:         if $S_i > S_i^{max}$ then
21:             $A_i = A_i - (S_i - S_i^{max})$
22:         else
23:             $A_i = min\{A_{max}, A_i + cool\_factor\}$
24:         end if
25:     end for
26: end while
27: return sum of load served at each time step
```

3.4. UPFC CONTROL

UPFCs are employed to ideally prevent cascading failures from occurring, or at least to slow their progression, providing system operators time to recover from the initial fault. UPFC set points are desired that maximize the time to the next line failure. This is equivalent to maximizing the equation

$$\min_{i \in Lines} \frac{A_i}{(S_i - S_i^{max})} \tag{4}$$

In this manner UPFCs are used to delay a cascade, if not prevent it entirely, providing time for system operators to remedy the situation.

SQP is used to determine UPFC points which maximize Equation (4). SQP makes successive calls to a steady state loadflow to determine the time to next line failure for different UPFC set points. SQP is used to find UPFC set points for both real and reactive power. Thus, for a system with k UPFCs, there are $2k$ unknowns to be determined.

In preliminary tests, the UPFC control heuristic was shown capable of, on average, serving more load than the no UPFC system under a wide range of faults; however, for a little over 10% of the random UPFC placement/fault pairs tested the UPFC set points determined by this heuristic changed the lines which fail during the cascade and performed worse than the no UPFC system. To compensate for such cases, the UPFC control algorithm is allowed to place all the UPFC devices in bypass mode, where they do not affect the line on which they sit. If it is determined that the UPFC control heuristic will cause a change in the resulting cascade which forces less load to be served, then the UPFCs are placed in bypass mode. To determine when to place the UPFCs in bypass mode the UPFC control algorithm performs two simulations: one with the control heuristic, and one with the UPFCs in bypass mode.

3.5. EXPERIMENTAL SETUP

The co-optimization versions of simulated annealing (CoSA) and gradient ascent (CoGA) as well as the canonical coevolutionary algorithm, with (I-CoEA) and without islands (C-CoEA), are employed to demonstrate the effectiveness of the proposed methodology. C-CoEA, I-CoEA, CoSA and CoGA are used to find UPFC placements for a twenty-five bus, forty-one line, subset of a heavily stressed version of the IEEE 118 bus test system. Figure 3.1 shows a diagram of the IEEE 118 bus system. The subset of the system used in the tests presented here corresponds to *Area 2* in the diagram. Refer to Appendix for the line capacities used in the experiments presented here.

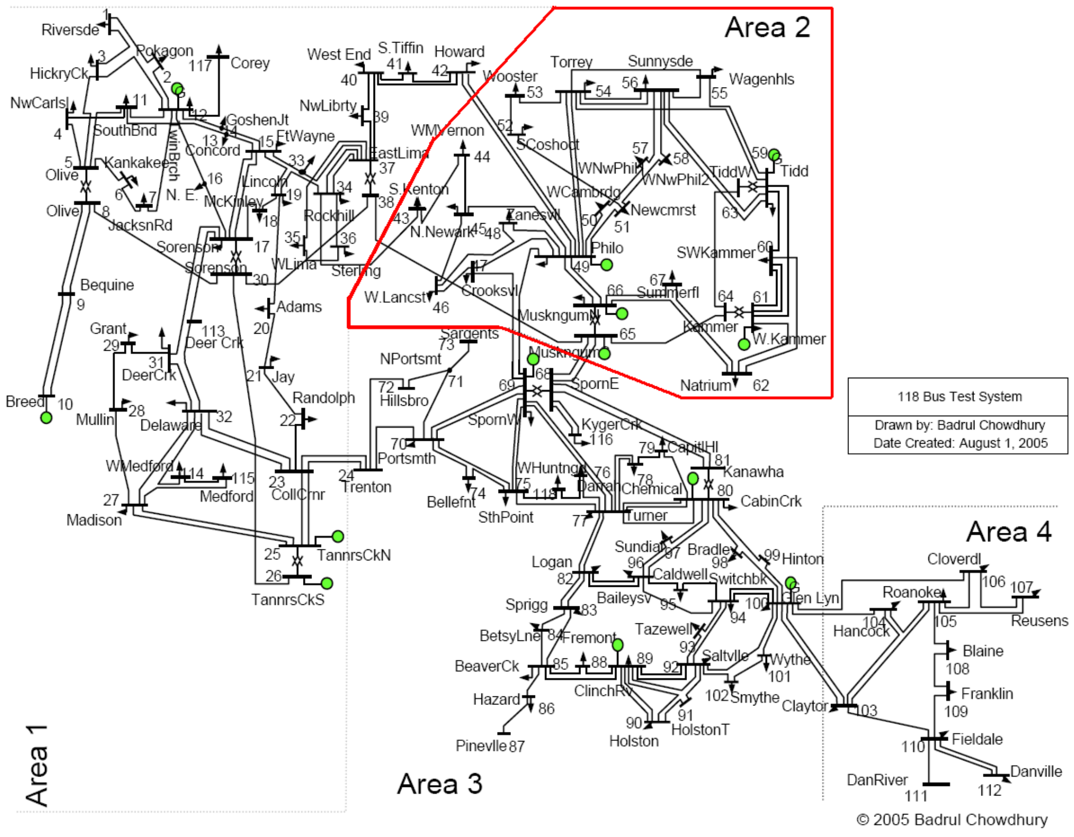


Figure 3.1. IEEE 118 bus diagram.

The cost incurred by a fault, f, with a hardening, h, in place, $L(h, f)$, is assumed to be directly proportional to the demand which can not be delivered. Equation (1) and Equation (2), the expected cost of a hardening h and failure f, respectively, are simplified by assuming the initial cost of all hardenings/placements to be zero and the expected number of times a given fault will occur is a function of the number of lines which fail, represented by $|f|$. For a given fault, f, E_f is defined to be $\frac{1}{1+|f|^2}$. Thus, any simultaneous fault of $|f|$ lines is assumed to occur, on average, $\frac{1}{1+|f|^2}$ times throughout the lifetime of the system, and the number of times a fault is expected to occur decreases roughly as the reciprocal of the square of the number of lines outaged during the fault. $|f|$ includes only those lines outaged initially by the fault and does not include additional line failures caused as a result of the initial outages (i.e., an initial fault consisting of the outage of two lines but inducing a cascade which causes the outage of an additional three lines will still be expected to occur $\frac{1}{1+2^2}$ times).

Placements are evolved to maximize the negation of Equation (1) and faults are evolved to maximize Equation (2).

Table 3.1 details the parameters used for each co-optimization algorithm.

3.6. RESULTS

As it is infeasible to evaluate a placement against all 2^{41} possible faults, the best placements found by each co-optimization algorithm are compared based on their performance over all single and double line contingencies. Removal of three lines, from the forty-one line test system, represents, on average, removal of a little more than seven percent of the system's transport capacity. Such a large initial outage is very unlikely to occur in a real-world system, as such comparing against all single and double line contingencies provides a fair estimate of the true, objective, quality of the placements. However, it is important to note that the resulting cascade induced by the initial contingencies is not limited to two lines. Thus, the single and double line contingencies could result in cascades of more than two lines.

As a base line by which to compare the effectiveness of the co-optimization algorithms for CIP, a uniform random sample of 125 UPFC placements of between one and ten UPFCs is taken to provide an estimate of the average UPFC placement effectiveness. Table 3.2 provides the results averaged over all 125 random placements. The random UPFC placements were, on average, able to serve about 3% more system demand than the no UPFC system.

Table 3.3 shows the results of the best placement found by the C-CoEA, I-CoEA(3), CoSA and CoGA co-optimization algorithms against all single and double line contingencies.

Without UPFCs in the system, only 85.42% of the demand was served, averaged over all single and double line contingencies. The placements found with the co-optimization algorithms served about 8% more demand than otherwise would have

Table 3.1. Parameters for C-CoEA, I-CoEA, CoSA, and CoGA.

Parameter	Value
Population size	15
CoEA recombination	One point crossover
UPFC population mutation	If there are no UPFC's in the system then a UPFC is randomly added. Otherwise 25% of the time a UPFC is added or removed from the system (with equal probability), and the other 75% of the time a UPFC is moved from the line it is currently on to another line in the system.
Fault population mutation	If there are no outaged lines then an outaged line is randomly added. Otherwise 25% of the time an outaged line is added or removed from the system (with equal probability), and the other 75% of the time an outaged line is moved to a new line.
CoGA number of evaluated neighboring individuals	1
CoEA offspring	15
CoEA parent and survival selection	Binary tournament selection
CoSA cooling schedule	Linear
I-CoEA number of islands	3
Interaction function evaluations	1,350,000
Number of runs	30
UPFC population initialization	Placement of a UPFC on each line in the system with 5% probability.
Fault population initialization	Outage of each line in the system with 5% probability.

been served. However, the only statistically significant differences, with $\alpha = 0.05$ on a two-tailed t-test assuming unequal variances, were that both C-CoEA and I-CoEA(3) outperformed CoGA.

The placements from all three co-optimization algorithms were, on average, superior to the randomly generated placements, by serving about 5% more demand,

Table 3.2. Random UPFC placements.

Ave. Demand Served(Std. Dev.)	Ave. Num. of UPFCs(Std. Dev.)
88.65%(1.93)	4.91(1.59)

Table 3.3. Co-Optimization algorithm effectiveness.

Algorithm	Ave. Demand Served(Std. Dev.)	Ave. Num. of UPFCs(Std. Dev.)
C-CoEA	94.10%(1.76)	5.13(2.03)
I-CoEA(3)	93.96%(1.32)	4.87(1.59)
CoSA	93.49%(1.56)	6.03(1.69)
CoGA	93.29%(1.10)	5.37(1.35)

demonstrating how the co-optimization methodology can produce UPFC placements capable of minimizing the damange caused by likely faults.

3.7. DISCUSSION

This section presented a novel methodology for CIP employing co-optimization to simultaneously discover effective system hardenings and damaging system faults. The system hardenings are optimized to effectively withstand the current set of faults and the system faults are optimized to challenge and thus promote further advances in the system hardenings.

The effectiveness of this methodology was demonstrated on a real-world power transmission system case study. A population of UPFC device placements was optimized simultaneously with a population of line outages. The results of this study show that co-optimization algorithms are able to discover placements of UPFC devices capable of increasing system resilience to probable faults.

Unlike the experiments presented in Section 2, C-CoEA performed as well as both CoSA and CoGA in the CIP setting. This suggests that C-CoEA does not exhibit any of the studied pathologies in this setting, allowing it to perfom on par with I-CoEA(3), CoSA and CoGA, or that the UPFC placement and fault fitness landscapes are well suited for evolutionary methods. This second possibility is consistent with the author's previous findings in [17], where experimental results on optimizing UPFC placements against a fixed set of faults show evolutionary algorithms to be superior to simulated annealing.

An avenue of future work is to compare the performance of the best UPFC placements found by C-CoEA, I-CoEA(3), CoSA and CoGA over all single and double line contingencies, to the performance of UPFC placements evolved against all single and double line contingencies using traditional evolutionary computing. This will strengthen the results presented in this section by providing a comparison between UPFC placements found by co-optimization techniques and UPFC placements optimized against the set of all single and double line contingencies.

The methodology presented in this section can be applied to hardening arbitrary infrastructures as well as to hardening systems of interconnceted infrastructures. As cascading failures now threaten to jump infrastructure borders, hardening a single infrastructure may no longer be sufficient. Interactions between infrastructures need to be taken into consideration.

4. NO-FREE-LUNCH: A THEORETICAL FOUNDATION

The No-Free-Lunch (NFL) theorem is a fundamental result in the field of black-box function optimization. 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 interactive nature of fitness evaluation in co-optimization, co-optimization algorithms are not, in general, a member of this class [18].

Recent work has shown that certain classes of coevolution, and therefore also co-optimization, exhibit free lunches. As such, any framework designed to perform NFL like analysis for co-optimization should allow for the discussions of NFL like results for natural classes of co-optimization algorithms.

In this section a novel framework for analyzing NFL like results for co-optimization which satisfies this requirement is presented. The framework presented in this thesis classifies co-optimization algorithms by the solution concept which they implement (i.e., the type of solution they were designed to find).

4.1. BACKGROUND

NFL like questions have been discussed in both the traditional function optimization and the coevolutionary settings. Here an overview of previous related NFL work for both traditional function optimization as well as for coevolution is provided. During this overview the two models of search algorithms used in NFL work as well as in the co-optimization setting are presented.

4.1.1. Traditional Optimization. Here a brief overview of the framework used in the NFL theorem for traditional black-box function optimization is presented. The reader is referred to [36] for the original and full presentation of the NFL theorem.

Let $\mathcal{F} = \mathcal{Y}^{\mathcal{X}}$ be the set of all $|\mathcal{Y}|^{|\mathcal{X}|}$ cost functions from a finite search space, $\mathcal{X}$, to a finite set of possible cost values, $\mathcal{Y}$. As computer memory is inherently finite, the restriction to a finite search space is not limiting (i.e., the search space of real numbers is still restricted to those numbers representable by some finite number of bits). However, it has been shown that in continuous domains the standard NFL theorem does not hold [37].

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 or 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 [36].

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 [36]:

$$a : d_m \in \mathcal{D}^* \rightarrow \{x | x \notin d_m^x\} \tag{5}$$

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 and is referred to for the remainder of this thesis as the *traditional model.* See Figure 4.1 for a visual representation of a search algorithm[3].

For simplicity in this thesis a search algorithm is considered to be completely deterministic[4]. The reader is referred to [36, 18] for details on how to extend this to stochastic search algorithms, where the next chosen point is a random variable in a probability distribution conditional upon the observed sequence.

[3]This figure is a recreation of the one used in [38]

[4]Instances of stochastic algorithms may be viewed as deterministic when used with a specific pseudo-random number generator with a given seed [36].

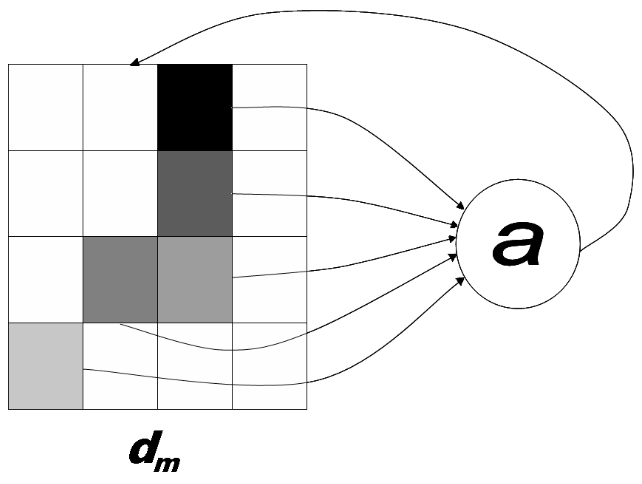


Figure 4.1. Data driven algorithm. Data points are sampled from a finite search space (left) and a new unique search space point is choosen based upon the observed data points.

The original 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[5].

Theorem 4.1 (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:*

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

where $\mathcal{F}$ is the set of all cost functions from a finite search space $\mathcal{X}$ to a finite set of cost values $\mathcal{Y}$.

This result has since been generalized to subsets of $\mathcal{F}$ which are closed under permutation, rather than $\mathcal{F}$ as a whole [39].

A search algorithm's performance is based upon the sequence of cost values it observes. Formally, a performance metric is a function mapping sequences of costs

[5]The statement of the original NFL theorem is left in terms of probabilities even though only deterministic search algorithms are considered in the analysis presented in this thesis.

values to a real number indicating how well an algorithm, which observed that sequence of cost values, performed:

$$\Phi : d^y_m \in D^* \rightarrow \Re$$

For example, a performance metric might be the last observed cost value, or the greatest cost value seen so far. Thus, algorithms are compared upon the distinct sequences of cost values they observed. Factors such as wall time, the amount of time the algorithm takes to run, fall outside the range of the NFL theorems.

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 equally well when averaged over all cost functions under any performance measure.

Corollary 4.2 (No Free Lunch)**.** *For any two search algorithms a and b, any positive integer m, any sequence of m cost values d^y_m and any performance metric Φ:*

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

4.1.2. Coevolutionary Free Lunches. Free lunches exist in general in coevolution [18], and thus also in co-optimization. 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 desired. 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 (5)), 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 interactions 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:

$$A : D^* \rightarrow P_1 \tag{6}$$

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 (6)) is the second search algorithm model employed and is further referred to as the *candidate selection model.*

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 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 [18] 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, P_2. 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 $\min_{O \in P_2} g(C, O)$ [18].

The framework presented in this thesis approaches the question of the existence of free lunches in CoEAs from the point of view of solution concepts and the weak preference relation. Informally, a given solution concept exhibits no free lunches under the weak preference relation 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.

Before formalizing this framework it is shown how a solution concept and the weak preference relation induce a directed acyclic graph (DAG) on the set of possible solution configurations. Search algorithms are then modeled in terms of this induced preference DAG.

4.2. PREFERENCE DAG

The weak preference relation induces a directed graph on the space of solution configurations, $C(P_1 + \cdots + P_n)$. An edge is defined to be from configuration β to configuration α when $\alpha \succ_W \beta$. That is, edges point in the direction of increased preference and may be thought of as routes through configuration space to the global solutions. Algorithms are desired 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 (that is a DAG).

Given a solution concept, each interaction function induces a different preference DAG. The preference DAG induced by an interaction function g is denoted by $Pref(g)$.

4.3. ALGORITHM MODELS

Here both the traditional model and the candidate selection model used in NFL literature are modeled in terms of preference DAGs.

4.3.1. 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 (5)), 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:

$$(x_1, y_1), \cdots, (x_m, y_m)$$

(where exploration means examination under the interaction function, i.e., fitness evaluations). The visiting of a point x_i in the search space can be viewed 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.

4.3.2. 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 [18] 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:

$$A : D^* \to C(P_1 + \cdots + P_n) \tag{7}$$

This is a generalization of the champion selection function, Equation (6) [18]. It is assumed 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 [11], 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 arbitrary combinations of search heuristics and candidate selection functions can be compared, this thesis restricts its attention to a fixed, but arbitrary, candidate selection function. This is done because allowing arbitrary combinations, or even a fixed, but arbitrary, search heuristic 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 the number of data points, m, is allowed to 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.

4.4. PERFORMANCE METRICS

The weak preference relation provides a means by which to measure performance in co-optimization. Algorithms which consistently find more preferred solution configurations than other algorithms are desired.

Formally, a performance metric is defined to be a function Φ 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, Φ, if two labeled graphs $Pref(g)$

and $Pref(g')$ are isomorphic, then $\Phi(Pref(g)) = \Phi(Pref(g'))$. This fact is exploited 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 preferred to. Thus, under the candidate selection model, if the solution configuration selected by an algorithm A is preferred to more nodes than the solution configuration selected by an algorithm B then the performance of A is superior to that of B in that instance.

Intuitively this definition makes sense as only the relative ordering, under the weak preference relation, of the visited solution configurations differentiate between algorithm performance.

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

4.5. NO-FREE-LUNCH DEFINITIONS

Informally, free lunches do not occur under the weak preference relation only when for all search algorithm pairs a and b, any performance metric value, V, and any performance metric Φ, Φ 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'.

4.5.1. 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 4.3 (Traditional Model - NFL). *A solution concept is said to not exhibit free lunches under the weak preference relation 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:*

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

where $Pref(a_g^m(\emptyset))$ denotes the labeled graph induced by algorithm a after m iterations on interaction function g[6].

In other words, Definition 4.3 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 $\mathcal{F}(g)$. If such a bijection $\mathcal{F}$ exists, for any performance metric, Φ, it follows that

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

for all $g \in \mathcal{G}$. Thus, every possible performance value occurs the same number of times for any pair of algorithms when ranged over all possible interaction functions.

4.5.2. Candidate Selection Model. For the more general search algorithm model just the solution configuration produced after m iterations of the search heuristic is considered, as was done in [18]. As with the traditional model, the solution configuration returned by the candidate selection function is considered 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.

Definition 4.4 (Candidate Model)**.** *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 $\mathcal{F} : \mathcal{G} \to \mathcal{G}$ such that:*

$$\forall g \in \mathcal{G} \;\; Pref(A(a_g^m(\emptyset))) \simeq Pref(A(b_{\mathcal{F}(g)}^m(\emptyset))) \tag{8}$$

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

[6] In reality this also depends on the solution concept used; however, this relationship will largely be left implicit

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.

Note that the bijection $\mathcal{F}$, for all values of m, is not required to produce interaction functions on which algorithm b produces an isomorphic graph to that of a. In general $\mathcal{F}$ will be a function of m, with different values of m producing different bijections.

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

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

for any performance metric Φ, and all $g \in \mathcal{G}$.

The NFL definition for the candidate selection model deals with the informativeness of search heuristics. That is, if a solution concept and candidate selection function satisfy Definition 4.4, 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.

4.6. DISCUSSION

This section presented a framework for discussing No-Free-Lunch like results for classes of co-optimization. Co-optimization algorithms are naturally classified by the solution concept which they implement, and algorithm performance is based upon labeled graphs induced by the algorithm and the weak preference relation.

The weak preference relation depends only on the contexts in which solution configurations appear as solutions and not on the properties of the solution concept used. This non-dependence on the particular solution concept used results in a loss of discriminating ability. As an example consider the solution concept of MaxiMin. A natural measure of the quality of a solution configuration is the minimum value it obtains over all test cases. However, such a measure is not expressible with the weak preference relation. While $\alpha \succ_W \beta$ implies that $\min_{t \in \mathcal{T}} g(\alpha, t) > \min_{t \in \mathcal{T}} g(\beta, t)$ the reverse is not true.

While the framework presented in this section employs the weak preference relation as a means to compare algorithm performance, measures of preference other than the weak preference relation may be used in the same manner. For example, a preference relation defined by

$$\alpha \succ_{MaxiMin} \beta \text{ iff } \min_{t \in \mathcal{T}} g(\alpha, \tau) > \min_{t \in \mathcal{T}} g(\alpha, t)$$

could be used in place of the weak preference relation to induce a preference DAG. The preference relation $\succ_{MaxiMin}$ naturally applies to the MaxiMin solution concept, but not to maximization of expected utility. Also since $\alpha \succ_W \beta$ implies $\alpha \succ_{MaxiMin} \beta$ but the reverse is not true, $\succ_{MaxiMin}$ provides a greater amount of distinguishing ability between solution configurations for the MaxiMin solution concept.

This provides a way in which to employ this co-optimization framework for other measures of solution configuration preference and presents a possible extension to this framework.

5. CONCLUSIONS

The primary contribution of this thesis is the introduction of the class of co-optimization algorithms. This class provides a natural generalization of coevolution to arbitrary optimization techniques and employs the same interactive fitness environment as does coevolution. Three co-optimization algorithms are introduced, in addition to coevolution, and are shown to be effective optimization techniques. It is shown that non-population based co-optimization algorithms naturally provide a greater amount of diversity amongst the individuals. As lack of diversity is a common problem in coevolution, with many additions having been suggested to alleviate it, non-population based co-optimization promises to be an effective optimization technique. Furthermore, the effects of disengagement are investigated on each co-optimization algorithm on the Compare-On-All test problem. It is observed that the non-population based algorithms, and the island based CoEA, are capable of reengaging and making progress even when initialized to be completely disengaged, and it is hypothesized that non-population based co-optimization algorithms are less susceptible to the effects of disengagement.

A novel co-optimization methodology for Critical Infrastructure Protection is presented along with a real-world case study. Co-Optimization is employed to simultaneously optimize hardenings for an electric power transmission system, in the form of placements of Unified Power Flow Controllers (UPFCs), and system faults, in the form of line outages. The co-optimized UPFC placements are compared over all single and double line contingencies and found to have discovered system hardenings capable of lessening the impact of a wide range of likely faults.

Finally, a novel framework for discussing No-Free-Lunch (NFL) like results for co-optimization is presented. While free lunches have been shown to exist in general in coevolution, and thus also in co-optimization, the framework presented in this thesis allows such questions to be phrased in terms of the type of solution desired (i.e., the solution concept employed). This allows co-optimization algorithms to be naturally classified by the solution concept they implement and NFL like results to be proven for such classes of co-optimization.

The generalization of coevolution to co-optimization presented in this thesis opens the door to matching interactive problem domains to the optimization technique best suited for that domain. For example, if a particular interactive domain exhibits one of the common pathologies under coevolution then the use of evolutionary computation may be swapped with another optimization technique which does not exhibit those pathologies in the given domain.

6. FUTURE WORK

The generalization of coevolution to co-optimization opens the door to many applications of co-optimization algorithms to problems previously only solved by coevolution. An interesting avenue of future work is to identify which co-optimization algorithms are best suited for which interactive problem domains. Such results would vastly improve the ease of use of co-optimization algorithms as time would not need be expended trying to identify the best algorithm for the problem domain of interest.

To further demonstrate the effectiveness of the co-optimization methodology for Critical Infrastructure Protection (CIP) presented in this thesis, the methodology needs to be applied to additional infrastructures and needs to incorporate multiple forms of system hardenings (i.e., topology changes as well as flow controllers). Further, this methodology can be employed to harden systems of infrastructures rather than only single infrastructures.

A possible extension to the NFL framework presented in Section 4, is the use of notions of solution configuration preference other than the weak preference relation. For individual solution concepts there are algorithm performance metrics which are not expressible in terms of the weak preference relation. For example, a preference relation defined by

$$\alpha \succ_{MaxiMin} \beta \text{ iff } \min_{t \in \mathcal{T}} g(\alpha, \tau) > \min_{t \in \mathcal{T}} g(\alpha, t)$$

naturally applies to the MaxiMin solution concept and could be used in place of the weak preference relation to induce a preference DAG. While $\alpha \succ_W \beta$ implies $\alpha \succ_{MaxiMin} \beta$, the reverse is not true. Thus, in a sense, the preference relation $\succ_{MaxiMin}$ provides a greater level of distinguishing ability than does $\succ_W$. However, while $\succ_W$ is solution concept independent, $\succ_{MaxiMin}$ applies only to the solution concept of MaxiMin and not to, for example, the maximization of expected utility solution concept.

The NFL framework presented in this thesis might be extended by allowing different preference relations to be employed to induce preference DAGs. Under each preference relation different performance metrics would be expressible. Such an extension might provide a means by which to classify performance metrics as well as to assess to what extent free lunches occur, for solution concepts for which NFL like results do not hold, by the characteristics which preference relations need to have in order for free lunches to occur under them.

APPENDIX

Line Ratings, S_{max} values, for Area 2 of the IEEE 118 Bus System

From Bus	To Bus	Line Rating
43	44	0.388033
44	45	0.758944
45	46	0.830441
45	49	0.772015
46	47	0.472097
46	48	0.357073
47	49	0.524914
48	49	0.529905
49	50	0.829812
49	51	1.045510
49	54	0.799406
49	54	0.799406
49	66	2.708691
49	66	2.708691
50	57	0.555417
51	52	0.438542
51	58	0.432726
52	53	0.262565
53	54	0.467182
54	55	0.262466
54	56	0.431543
54	59	0.469334
55	56	0.499582
55	59	0.534541
56	57	0.370952
56	58	0.228730
56	59	0.721000
56	59	0.721000
59	60	0.993488
59	61	1.188832
59	63	2.491195
60	61	1.690491
60	62	0.616106
61	62	0.652729
61	64	1.699646
62	66	0.614703
62	67	0.423885
63	64	2.491195
64	65	2.917117
65	66	1.636710
66	67	0.848251

BIBLIOGRAPHY

[1] Stephane Gerbex, Rachid Cherkaoui, and Alain Germond. Optimal Location of Multi-Type FACTS Devices in a Power System by Means of Genetic Algorithms. *IEEE Transactions on Power Systems*, 16(3):537–544, August 2001.

[2] Agoston Eiben and James Smith. *Introduction to Evolutionary Computing.* Springer, 2003.

[3] Kenneth De Jong. *Evolutionary Computation: A Unified Approach.* The MIT Press, 2006.

[4] Daniel Hillis. Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure. *Physica D Nonlinear Phenomena*, 42(1–3):228–234, June 1990.

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

[6] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2003.

[7] Scott Kirkpatrick, Jr. C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, May 1983.

[8] Edwin De Jong. The Maxsolve Algorithm for Coevolution. In *Proceedings of the 7th annual Genetic and Evolutionary Computation Conference*, pages 483–489, June 2005.

[9] Edwin De Jong. Objective Fitness Correlation. In *Proceedings of the 9th annual Genetic and Evolutionary Computation Conference*, pages 440–447, July 2007.

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

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

[12] Anthony Bucci and Jordan Pollack. Thoughts on Solution Concepts. In *Proceedings of the 9th annual Genetic and Evolutionary Computation Conference*, pages 434–439, July 2007.

[13] Sevan Ficici. Monotonic Solution Concepts in Coevolution. In *Proceedings of the 7th annual Genetic and Evolutionary Computation Conference*, pages 499–506, June 2005.

[14] Frans A. Oliehoek, Edwin 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, July 2006.

[15] Travis Service and Daniel Tauritz. Co-Optimization Algorithms. *To Appear in Proceedings of the 10th annual Genetic and Evolutionary Computation Conference*, July 2008.

[16] Travis Service, Daniel Tauritz, and William Siever. Infrastructure Hardening: A Competitive Coevolutionary Methodology Inspired by Neo-Darwinian Arms Races. In *Proceedings of 31st Annual IEEE International Computer Software and Applications Conference*, pages 101–104, July 2007.

[17] Travis Service and Daniel Tauritz. Increasing Infrastructure Resilience through Competitive Coevolution. *Accepted for publication in New Mathematics and Natural Computation.*

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

[19] Travis Service and Daniel Tauritz. A No-Free-Lunch Framework for Coevolution. *To Appear in Proceedings of the 10th annual Genetic and Evolutionary Computation Conference*, July 2008.

[20] Josh Bongard and Hod Lipson. 'Managed Challenge' Alleviates Disengagement in Co-evolutionary System Identification. In *Proceedings of the 7th annual Genetic and Evolutionary Computation Conference*, pages 531–538, June 2005.

[21] Edwin De Jong and Jordan Pollack. Learning the Ideal Evaluation Function. In *Proceedings of the 5th Annual Genetic and Evolutionary Computation Conference*, pages 277–288, June 2003.

[22] Edwin De Jong and Anthony Bucci. DECA: Dimension Extracting Coevolutionary Algorithm. In *Proceedings of the 8th annual Genetic and Evolutionary Computation Conference*, pages 313–320, July 2006.

[23] Melanie Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, 1997.

[24] William Siever, Daniel Tauritz, and Ann Miller. Blueprint for Iteratively Hardening Power Grids Employing Unified Power Flow Controllers. In *Proceedings of IEEE SoSE 2007–the 2nd International Conference on System of Systems Engineering*, pages 1–7, April 2007.

[25] U. S. DOE. Initial Blackout Timeline of the August 14, 2003 Outage, 2003.

[26] U.S.-Canada Power System Outage Task Force. Final report on the August 14th blackout in the United States and Canada: Causes and recommendations. Technical report, U.S. Department of Energy, April 2004.

[27] Ted Lewis. *Critical Infrastructure Protection in Homeland Security Defending a Networked Nation.* John Wiley & Sons, Inc., 2006.

[28] H.A. Abdelsalam, G.A.M. Aly, M. Abdelkrim, and K.M. Shebl. Optimal Location of the Unified Power Flow Controller in Electrical Power Systems. In *Proceedings of Power Systems Conference and Exposition*, volume 3, pages 1391– 1396. IEEE, 2004.

[29] Zhuo Lu, M.S. Li, W.J. Tang, and Q.H. Wu. Optimal Location of FACTS Devices by A Bacterial Swarming Algorithm for Reactive Power Planning. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 2344–2349, 2007.

[30] Narain Hingorani and Laszlo Gyugyi. *Understanding FACTS: Concepts and Technology of Flexible AC Transmission Systems.* IEEE Press, 2000.

[31] Adam Lininger, Bruce McMillin, Mariesa Crow, and Badrul Chowdhury. Use of Max-Flow on FACTS Devices. In *Proceedings of the 39th Annual North American Power Symposium*, pages 288–294, September 30–October 2, 2007.

[32] William Siever, Daniel Tauritz, and Ann Miller. Improving Grid Fault Tolerance by Optimal Control of FACTS Devices. *International Journal of Innovations in Energy Systems and Power*, 2(1):44–49, June 2007.

[33] Radha Kalyani, Mariesa Crow, and Daniel Tauritz. Optimal Placement and Control of Unified Power Flow Control devices using Evolutionary Computing and Sequential Quadratic Programming. In *Proceedings of the 2006 IEEE PES Power Systems Conference & Exposition–PSCE2006*, pages 959–964, October 29–November 1, 2006.

[34] John Chaloupek, Daniel Tauritz, Bruce McMillin, and Mariesa Crow. Evolutionary Optimization of Flexible AC Transmission System Device Placement for Increasing Power Grid Reliability. In *Proceedings of FEA 2005, the 6th International Workshop on Frontiers in Evolutionary Algorithms*, pages 516–519, July 2005.

[35] Thomas Overbye. Computation of a Practical Method to Restore Power Flow Solvability. *IEEE Transactions on Power Systems*, 10(1):280–287, February 1995.

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

[37] Anne Auger and Olivier Teytaud. Continuous Lunches are Free! In *Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference*, pages 916–922, July 2007.

[38] Mario Koppen, David Wolpert, and William Macready. Remarks on a Recent Paper on the “No Free Lunch” Theorems. *IEEE Transactions on Evolutionary Computation*, 5(3):295–296, 2001.

[39] 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, December 2004.

VITA

Travis Service was born on February 24, 1985 in Berkeley, California. He received his high school diploma and an Associate of Science degree from the Missouri Academy of Science, Mathematics and Computing in May of 2003 at Northwest Missouri State University. In the fall of that year, he enrolled in the Computer Science department and the Mathematics department of the University of Missouri - Rolla, now the Missouri University of Science and Technology, from which he received his BS in Computer Science and Applied Mathematics in December of 2005 and 2007, respectively. He enrolled in the Computer Science graduate program in the spring of 2006 and received his Master's degree in the May of 2008.