An Analysis of Cooperative Coevolutionary Algorithms

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

R. Paul Wiegand
Bachelor of Science, Computer Science
Winthrop University, 1996
Master of Science
University North Carolina Charlotte, 1999

Director: Kenneth A. De Jong,
Professor, Department of Computer Science

Fall Semester 2003
George Mason University
Fairfax, Virginia

DEDICATION

To my wife Andrea, whose considerable counsel was only a small part of her contribution towards my success in completing this dissertation.

ACKNOWLEDGEMENTS

I want to thank my committee members: Dr. Sauer, Dr. Hamburger, Dr. Luke, and Dr. De Jong. In particular, I would like thank Ken De Jong, whose advice and guiding hand has been a great source of help and unyielding service to me. Nearly all I know of Method comes from Ken and Descartes. If I master Method half as well as either, I will consider myself very fortunate. I will also single out Sean Luke, whose constant contrariness contributed to this work being about more than mere nasal membrane (I hope).

The next big group of acknowledgments must go to my fellow lab mates. Their considerable advice, support, and influence ranks among the most profound aspects of my education. In particular, I thank Liviu Panait, Jeff Bassett, Jayshree Sarma, and Bill Liles. Additionally, I would like to thank Thomas Jansen, as well as the entirety of Ingo Wegener's chair at Dortmund University. Their helpful influence is clear in the first half of my dissertation, but the extent of this influence cannot possibly be fully visible.

Because they deserve it and are not thanked nearly enough, I would also like to thank the staff of the Department of Computer Science at George Mason University. They are great group of people, who have always gone out of their way to help make this process a little easier for me. Likewise, I thank the people at Library and Information Services. They do a huge and irreplaceable service for those of us in deep study, and probably cannot be thanked enough.

On a personal level, I thank my wife for her support. She gives me strength and self confidence. In exchange, I wash the clothes. I get the better end of that deal, I think. Finally, I want to be among the many who say that life is an adventure of learning and for every question answered, there are a dozen more unanswered questions that are uncovered. Far from being discouraging, this truism is one for which I was prepared...and one of the reasons I feel that life is worth living. This preparation, and this love of learning, I owe to my parents, who definitely cannot be thanked enough.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

AN ANALYSIS OF COOPERATIVE COEVOLUTIONARY ALGORITHMS

R. Paul Wiegand
George Mason University, 2003
Thesis Director: Dr. Kenneth A. De Jong

Coevolutionary algorithms behave in very complicated, often quite counterintuitive ways. Researchers and practitioners have yet to understand why this might be the case, how to change their intuition by understanding the algorithms better, and what to do about the differences. Unfortunately, there is little existing theory available to researchers to help address these issues. Further, little empirical analysis has been done at a component level to help understand intrinsic differences and similarities between coevolutionary algorithms and more traditional evolutionary algorithms. Finally, attempts to categorize coevolution and coevolutionary behaviors remain vague and poorly defined at best. The community needs directed investigations to help practitioners understand what particular coevolutionary algorithms are good at, what they are not, and why.

This dissertation improves our understanding of coevolution by posing and answering the question: "Are cooperative coevolutionary algorithms (CCEAs) appropriate for static optimization tasks?" Two forms of this question are "How long do they take to reach the global optimum" and "How likely are they to get there?" The first form of the question is addressed by analyzing their performance as optimizers, both theoretically and empirically. This analysis includes investigations into the effects of coevolution-specific parameters on optimization performance in the context of particular properties of potential problem domains. The second leg of this dissertation considers the second form of the question by looking at the dynamical properties of these algorithms, analyzing their limiting behaviors again from theoretical and empirical points of view. Two common cooperative coevolutionary pathologies are explored and illustrated, in both formal and practical settings. The result is a better understanding of, and appreciation for, the fact that CCEAs are *not* generally appropriate for the task of static, single-objective optimization. In the end a new view of the CCEA is offered that includes analysis-guided suggestions for how a traditional CCEA might be modified to be better suited for optimization tasks, or might be applied to more appropriate tasks, given the nature of its dynamics.

Chapter 1

Introduction

1.1 Evolutionary Algorithms

Evolutionary algorithms (EAs) are heuristic methods for solving computationally difficult problems using biologically inspired notions of Darwinian evolution. They have been applied to a variety of problems, from static optimization to job-shop scheduling. EAs frequently have an advantage over many traditional local search heuristic methods when search spaces are highly modal, discontinuous, or highly constrained. As such they continue to be of great benefit for a large community of users with such needs.

Unfortunately, there are problems on which EAs tend to perform poorly, or for which no simple method for applying them is known. One such situation occurs when problems have very large search domains defined by the Cartesian product of two or more, interacting subspaces. For example, this is often the case when one would like to evolve some functional element in combination with its input data. In extreme settings, the space can be infinite, and some method of focussing on relevant areas is needed by the EA. Another situation in which it is difficult to apply an EA occurs when no intrinsic objective measure exists with which to measure the fitness of individuals. This can be the case when evolving game-playing strategies, for instance. Finally, when searching spaces of complex structures, EAs often have difficulties when no domain-specific modifications are made to help direct the search.

For these kinds of problems, researchers have turned to a natural extension of the evolutionary algorithm: coevolution. Coevolutionary algorithms have a lot of potential in terms of addressing the types of problems just mentioned. As such, they have become an important area of research in the field of evolutionary computation.

1.2 Coevolutionary Algorithms

As the reader will discover from the first few chapters of this work, the subject of coevolution is a complicated one. Researchers debate everything from pragmatic questions about the effectiveness of particular coevolutionary algorithms, to philosophical questions about what constitutes a coevolutionary algorithm in the first place. I will touch on many of these debates in the coming chapters, but perhaps it is best to start with a very high level answer to the basic question "what is a coevolutionary algorithm (CEA)?"

For now, the simplest answer is that a coevolutionary algorithm is an evolutionary algorithm (or collection of evolutionary algorithms) in which the fitness of an individual depends on the relationship between that individual and other individuals. Such a definition immediately imbues these algorithms with a variety of views differing from those of more traditional evolutionary algorithms. For example, one might favor the view that individuals aren't evaluated at all, but in fact their interactions are evaluated. Alternatively, one might look at individual fitness evaluation from the perspective of a dynamic landscape, given that the result of the evaluation is contextually dependent on the state of other individuals. In either case it is clear that they differ in profound ways from traditional EAs.

As we will see in the next couple of chapters, coevolutionary algorithms vary widely. The differences between cooperative and competitive coevolutionary algorithms are among the most fundamental distinctions. In the case of cooperative algorithms, individuals are rewarded when they work well with other individuals and punished when they perform poorly together. In the case of competitive algorithms, however, individuals are rewarded at the expense of those with which they interact. Though there may be many types of algorithms that fall into neither camp, most studies concern one or the other, and this dissertation is no exception: I focus almost entirely on cooperative coevolutionary algorithms (CCEAs). The reasons for this will become clear shortly.

1.2.1 The Hope of Coevolution

Coevolutionary algorithms offer a lot of hope to researchers and practitioners. At first blush they appear to have many advantages over traditional evolutionary methods. For example, there is some reason to believe they may be useful with very large problem spaces—infinite search spaces in particular. The hope is that CEAs will be able to focus the search on relevant areas by making adaptive changes between interacting, evolving parts. Coevolutionary algorithms also appear to have an advantage when applied to problems for which no intrinsic objective measure even exists. CEAs use subjective measures for fitness assessment, and as a result become natural methods to consider for problems like the search for game-playing strategies. Finally, in the case of cooperative algorithms in particular, there is the potential advantage of being natural for search spaces that contain certain kinds of complex structures, since search on the smaller components of the structure can be emphasized. I will discuss these all of these advantages in more detail in the next chapter.

Advantages like these lead researchers to view coevolutionary problem solvers as having great potential. For example, they are considered by many to be potentially quite powerful optimization tools. In the case of competitive methods, the attractive notion of an *arms race* encourages researchers. The idea is that continued minor adaptations in some individuals will force competitive adaptations in others, and these reciprocal forces will drive the algorithms to generate individuals with ever increased performance. A similar idea exists in the cooperative world, where parallel adaptive evolutionary forces help keep the algorithm driving along a (possibly infinite) gradient. Moreover, in the case of cooperative coevolution there is the hope that complex structures can be dynamically decomposed and solved in parallel.

Indeed, advantages like these have prompted practitioners to apply CEAs to a wide variety of problems. Optimization applications include the coevolution of fast, complete sorting networks (Hillis 1991) and the coevolution of maximal arguments for complex functions (Potter and De Jong 1994). Machine learning applications include using CEAs to search for useful game-playing strategies (Rosin and Belew 1996) and employing them to train dynamically structured recurrent neural networks (Potter and De Jong 2000). There are many more examples.

1.2.2 The Frustration of Coevolution

Despite the optimism that surrounds coevolutionary algorithms as potential problem solvers, they more often than not frustrate practitioners when they are applied. There are several practical disadvantages of these algorithms over traditional evolutionary approaches that become more clear in application. As we will see in the coming chapters, the dynamics of coevolutionary systems are often far more complicated than those of traditional evolutionary algorithms, as well as frequently quite counter intuitive. The algorithms tend to exhibit distinct and profound pathologies and are often much more sensitive to certain problem properties than their more traditional analogs. Moreover, the subjective nature of fitness makes measuring progress quite difficult in many cases.

The result of these disadvantages is that the algorithms often fail to perform well on even fairly “simple” problems. What is worse, the connection between parameter sensitivity and coevolutionary pathologies is not at all well understood. Further, coevolutionary researchers approach the field from different directions, applying terms differently or, more often than not, ambiguously. For all of these reasons, researchers have become increasingly more focussed on establishing some basic theoretical groundwork for understanding coevolutionary algorithms, but the gap between theory and practice is still quite wide.

The problem is that these early analyses have approached different aspects of coevolution from different perspectives, but no one has asked the most fundamental of all questions: “What do these algorithms *do*?”—or, if optimization is the goal, “Do they optimize?”

1.3 Understanding Cooperative Coevolution

My goal with this dissertation is to back up, study a simple class of algorithms in a simple context, and use a multilateral analytical strategy to attempt to answer the most basic questions about coevolution first. This section first motivates why such an understanding is needed, then describes my strategy and methodology for obtaining answers.

1.3.1 Motivation

The next chapter will make clear the degree to which coevolution has become increasingly the focus of analytical research. Still, theory for coevolutionary computation is in its infancy. The studies in the literature, discussed later in the dissertation, while certainly offering much in the way of promise for a greater understanding of coevolution, do not ultimately as yet provide much in the way of constructive advice for practitioners. What remains is to understand why this might be the case, how to change our intuition by understanding the algorithms better, and what to do about the differences. As is often the case, there is still a sizable gap between theory and practice of coevolution. What is specifically needed is to provide a better understanding to practitioners of what coevolution is good at, what it is not, and why.

The main point of this dissertation is to focus analysis of this question squarely at a subset of coevolutionary algorithms: cooperative coevolutionary algorithms. In so doing, the reader will see that cooperative coevolutionary algorithms have been frequently misapplied, and a *new view* of what they do should be considered by future researchers. In my final chapter, I will offer several specific high level suggestions for what such views might be.

1.3.2 Strategy and Methodology

This dissertation seeks to take a step back, look at coevolution in the simplest of possible settings from the perspective of an engineer that wants to use a tool, an engineer that wants to know: “What does this tool *do*?” or “For what is this tool useful?” I will call this question the “fundamental question of coevolution”. My method consists of taking this abstract question and refining it to more specific, answerable questions in particular contexts. The high level strategy is one that combines a variety of theoretical research tactics (primarily evolutionary game theory, combined with some run time analysis) with empirical component analyses of real, but simple CEAs on several levels. The goal is to make the gap between theory and practice smaller by attempting to address the specific forms of this fundamental question in appropriate contexts.

While competitive coevolution may in some sense be a more attractive prospect for study, given its historical emphasis and its intrinsic interest as a model in and of itself, it is also significantly more complicated than cooperative CEAs. Since the evidence that thus far exists suggests that even CCEAs can be quite complicated to understand, it seems prudent to start there. This work concentrates on answering one aspect of the fundamental question for cooperative coevolutionary algorithms: “Do cooperative coevolutionary algorithms optimize?”

In fact, one naive and simple answer to the fundamental question of *cooperative* coevolution is that the algorithms are *static optimizers*. They are, after all, very frequently applied to static optimization problems. But is this really the case? What does this even mean, given that virtually any heuristic could be applied as a type of optimization algorithm, whether appropriate to such a task or not? These are important and practical questions and, as a result, this question of whether or not CCEAs are static optimizers is the aspect of the fundamental question on which I will focus. Importantly, the central theme of the dissertation is to answer this question resoundingly that CCEAs are not, by nature, static optimizers. This theme will be expanded by first treating them *as* optimizers and analyzing them with respect to their performance on static optimization problems, re-posing the fundamental question as “how well do they optimize?” This view uncovers salient points about properties of problem domains, as well as the design choices affected by those properties. I then reverse the picture and offer theoretical and empirical evidence that the algorithms attempt to seek a form of "balance" in a variety of ways, and this may or may not correspond with an external notion of optimality (depending on the problem). In the end, it is my suggestion that practitioners either change the problems to which they apply CCEAs, or modify them such that they are more geared toward static optimization.

1.3.3 Contributions

This thesis contributes several key items to the field of coevolutionary computation (CoEC).

- The largest and, to date, most complete hierarchy of design choices available for coevolutionary algorithms is provided. This hierarchy, illustrated on page 23, is explicated in some detail in Chapter 3, offering a relatively thorough map with which practitioners and researchers may make more informed choices when constructing such algorithms.

- A much greater understanding of the relationship between problem decomposition and representational decomposition is provided. In so doing I dispel common misconceptions about the effects non-linear relationships have on search performance, and provide more constructive information about these relationships in terms of coevolutionary search. In addition, I provide analysis and advice for what kinds of interaction methods to use under these kinds of problematic conditions.

- Most importantly, the dissertation provides much needed insight into the role and purpose of cooperative coevolutionary algorithms. These algorithms are misapplied when tasked with static optimization. Indeed, researchers should reconsider such applications for these algorithms, or consider modifying the algorithms. In fact, the analysis in Chapter 5 helps us understand this conclusion, but also provides information for how to modify the algorithm in more directed and rational ways to accomplish research goals. Some example modifications are briefly discussed in Chapter 6.

In addition, I've attempted to make certain that my research follows a clear methodological framework, establishing a working model for how such analysis can be conducted in the future. This framework includes the following ideas.

- A high level question is presented at the start of the thesis, and this question is used to frame the entire work. More specific, researchable questions are generated by looking at the more basic question from different perspectives, and hypotheses are produced from these more specific questions.

- Instead of investing all of my effort in a single analytical tool, I consider a multilateral approach that combines several, very different formal methods. These methods, which include randomized algorithm analysis and dynamical systems analysis, are applied to different questions in order to elicit answers appropriate to those questions.

- The bridge between theory and practice is built with empirical research. Indeed, in the thesis there are many empirical studies for exactly this purpose; however, in all cases empirical research *follows* theoretical research. The theory is used to *guide* the empiricism.

1.4 Organization

The remainder of the dissertation is arranged as follows.

Chapter 2 provides brief but necessary overviews of evolutionary computation, as well as coevolutionary computation. A relatively detailed survey of analytical research in the field of coevolutionary computation offered. The result is high level background material helpful for providing context and foundations for understanding the rest of the dissertation.

Chapter 3 categorizes coevolution in a much more detailed way, provides detail regarding the CCEA architecture used as the basic algorithm for this research, and presents some notational and terminological definitions. The class of algorithms, as well as the motivation for using them will be clarified in this chapter.

Chapter 4 considers the presented CCEA model as a static optimizer. It begins by first describing how it can be applied to static optimization problems, then describes mechanisms of the model that allow it to possibly gain leverage over traditional EAs in some contexts, using both run time and empirical analyses to help the reader understand these advantages. The chapter then explores what is perhaps the most complicated and interesting distinguishing aspect of the CCEA from traditional EAs: the mechanisms of interaction required to assess fitness. Finally, the challenges facing those applying CCEAs to such problems are discussed in some detail. The result is a better understanding for how properties of the problem affect the choices design engineers must make with respect to the mechanics of how individuals interact for fitness. Several common myths about such properties are dispelled.

Chapter 5 reverses the picture entirely, rejecting the simple idea that CCEAs are intrinsically built for static optimization. I present a dynamical systems approach, using the tools of evolutionary game theory, in order to begin to describe some of the limiting behaviors of these algorithms. This theory suggests that the algorithms tend toward various forms of "balance" between populations, which may or may not have anything to do with optimality of the search space as design engineers think of it when they apply the algorithm to static optimization problems. It concludes by constructing specific and reasonable counter examples for real CCEAs, and demonstrating that the algorithms behave differently than one might expect in such circumstances.

The final chapter delivers the decisive message of my work: we must *change* the way we apply and study CCEAs in the future; a *new view* of these algorithms is required if we are to understand how they work, and when they will be successful. The chapter opens the door to follow-ups of the fundamental question. The responses offered here are to either change one's use of the algorithm, or change the algorithm itself. Both possibilities are briefly explored at a high level, making the demonstrable point that CCEAs are not frustratingly otiose tools, they are merely misapplied ones. I end with a conclusion discussing in more detail how my analysis contributes the coevolutionary computation community in general.

Chapter 2

Background

This chapter provides basic background material in order to give the reader some context for understanding the analysis discussed later. By the end of the chapter, the reader should have a broader conception of the field of coevolutionary computation, and the work that has been done to understand it.

This background chapter is organized as follows. The first sections are brief overviews of evolutionary coevolutionary computation. Since my work is analytical in nature, the third section is a survey of historical and contemporary analytical research into coevolutionary algorithms. I leave the specifics of the architecture on which my analysis centers for the next chapter.

2.1 Overview of Evolutionary Computation

The field of *evolutionary computation* (EC) is one that merges inspiration from biology with the tools and goals of computer science and artificial intelligence. The field offers powerful abstractions of Darwinian evolutionary models that allow for a wide range of applications from conceptual models of biological processes to technical applications in problem solving. Such methods have proven themselves to be interesting complex systems to study and, perhaps more importantly, often very robust problem solving mechanisms. Nature-inspired, yet honed and designed by engineers, algorithms based on EC (so-called *evolutionary algorithms* (EAs)) frequently demonstrate uncanny adaptive prowess when applied to difficult problems where search spaces have properties that make other, more traditional methods, less tenable (e.g., lack of continuity, high degrees of modality, highly constrained sub-spaces, etc.). The fascinating nature and often surprising successes of EC have drawn researchers to the field for the better part of half a century.

EAs are powerful heuristic methods for solving many types of computationally difficult problems. They typically draws their power from stochasticity and parallelism. While there are many different types of EAs, most EAs have their most basic elements in common: they generally begin with a population of potential solutions, make small changes to this population, and prefer changes that are objectively more “fit”. As such, they (hopefully) tend to “evolve” better solutions gradually over time. Regardless, all EAs share Darwin’s notion of “survival of the fittest” at some level, and it is precisely this property that engineers exploit in order to use these systems to solve problems.

An abstract evolutionary algorithm will be presented. What the algorithm abstracts are the major details that make a particular EA, but what it retains are the basic threads connecting all EAs: heredity, survival of the fittest, and at least some element of stochasticity. After discussing this abstraction, I will offer a short discussion on two very different canonical EAs in order to give the reader some perspective about the choices available to algorithm designers. Finally, I conclude the overview of EC by suggesting some of the ways in which these algorithms fail to serve practitioners, urging them toward extensions such as coevolution.

2.1.1 An Abstract Evolutionary Algorithm

Since there is evidence that some notion of EC has been known to parts of the computer science community for over 50 years (Fogel 1998), it should be unsurprising that many such algorithms exist. Nevertheless, a more modern view of EAs is a more unified view, concentrating on the similarity of these algorithms, rather than enumerating them as separate algorithms altogether (De Jong 2004; Michalewicz 1996). Indeed, it is the choices for how individuals in populations are represented, as well as how populations are formed, altered, and selected from that determine the specific type of EA being implemented. A generic form of a basic EA is shown below in Algorithm 1. This form will serve as a template for algorithms I discuss throughout this document. As should be clear shortly, the many types of existing (and still undiscovered) EAs are possible by specifying the various components of this general algorithm. First I present the pseudo-code for the algorithm (below), then I will discuss each of its major elements one by one.

Algorithm 1 (Abstract Evolutionary Algorithm).

```
1.  Initialize population of individuals
2.  Evaluate population
3.  $t := 0$
4.  do
        4.1 Select parents from population
        4.2 Generate offspring from parents
        4.3 Evaluate offspring
        4.4 Select survivors for new population
        4.5 $t := t + 1$
    until Terminating criteria is met.
```

Initialization

The first step in this algorithm, initialization, is often very simple and is tied very strongly to the representation choices for individuals made by the algorithm designers. Initialization can be done for a variety of reasons and in a variety of ways. For example, it can serve the function of refining samples that have already been discovered by some other search process; however, much more frequently initialization is random.

In random initialization, individuals are distributed randomly about the search space defined by the representation of the individuals. Such random initialization procedures offer the hope of providing the algorithm with diverse information about the search space.

Representation and Evaluation

To determine how evaluation is performed, design engineers must understand the problem domain and the representation chosen for this domain. Individuals are often thought of as *encoded* forms of potential solutions to some problem. As such, these potential solutions typically must be *decoded* somehow in order to measure their fitness in order to carry out evolution. This process of decoding and measurement is what is meant by *evaluation*, steps 2 and 4.3 of Algorithm 1.

While evolutionary algorithms have been applied to many kinds of problems, the most obvious and natural application is one of optimization. In such domains, individuals are encoded to represent potential arguments to the optimization problems, and evaluation consists of decoding these representations and determining fitness via some kind of objective assessment. By gradually refining potential solutions to such a problem, an EA is often able to produce increasingly more optimal solutions over time.

Two examples may serve to make this clearer. First, individuals may represent arguments for an optimization problem as binary strings. In such a case, individuals with this string-based representation may have to be decoded into real-valued arguments and given to the function in order to obtain the objective function value. Alternatively, individuals may represent the arguments directly as a vector of real values. Again, this vector must be extracted from the individual and given to an objective function in order to obtain fitness.

Selection

There are essentially two different kinds of selection that occur in an evolutionary algorithm: parent selection and survival selection. In either case, selection typically involves some kind of biased consideration of the fitness values assigned to individuals during evaluation, a preference toward more highly fit individuals.

Selection methods in EAs can vary widely, using one of many types of parent *or* one of many types of survival selection, but not necessarily (and not typically) both[1]. Selection methods can be very greedy (in the sense of fitness-bias) or quite mild, stochastic or deterministic. In some informal sense they can be seen as directors of the evolutionary path.

Representation and Genetic Operators

Retaining the notion of heredity from biological evolution suggests that offspring are like their parents on a genetic level, but not identical to them. The process by which offspring in an EA are constructed is one that involves the application of operators that, like the biological process, preserve this notion of heredity by transferring altered genetic material from parent to offspring. EC researchers refer to the operators responsible for generating offspring as *genetic operators,* and they often take the form of abstract of notions of *mutation* and *recombination* found in biological genetics. Like evaluation, genetic operators are tied intimately with an individual's representation.

In mutation, an offspring is altered from its parent (ostensibly due to an error while copying genetic material). In cases of binary representation, this may mean stray bits here and there being toggled from zero to one or vice-versa. With real-valued representations, this may mean randomly generated offsets in argument values. Whatever the specific mechanism and interpretation, the abstract idea is the same: an offspring's genetic material undergoes some change due to an outside force of some kind.

Recombination is somewhat different. Here two or more parent genes are combined to produce offspring with some traits of both (all) parents. For example, offspring represented by binary strings may be produced by combining substrings from two parents, while in real-valued representation offspring may be geometrically quite similar to both parents in terms of their Euclidian locations in the argument space.

In both cases, successful EA genetic operators tend to retain the notion of *heredity*: offspring are similar, but not identical, to their parents on a genetic level. One or both kinds of operators may be employed at varying levels in any given EA. Additionally, in both cases there is a vast number of such operators, tied to the representation chosen.

Termination

How one stops an EA varies as well, though there are some relatively simple, traditional criteria. For example, running for a fixed number of time steps (often called *generations*) is common. It is also not unusual to run the algorithm until the degree of change in the population has fallen below some threshold. The choice of when to stop (or often, when to re-start) an EA, coupled with choices of population sizes, often corresponds with how design engineers wish to split up the work-load of the search: how parallel the search should be, how many resources one wishes to invest in a given search, etc.

In analytical settings, one often is interested in running an algorithm until the first time the global optimum is reached. While this is usually an unhelpful stopping criterion in practical settings (since the global optimum is often unknown), answering theoretical questions about how long one expects to wait for such an event can be informative (Wegener 2002).

[1] An alternative viewpoint suggests that there are always both selection methods, but typically one is a uniform deterministic selection method.

2.1.2 Canonical EAs

The notion of what choices are available to EA designers at an abstract level should be clear now; however, it is from the instantiation of these choices that particular EA classes arise. These classes vary in how they can best be applied, as well as how well understood they are for particular domains. Theory for the different algorithms takes on quite different forms and is often very particular to representational and/or operator specifics. No one theory yet exists to unite them. Nevertheless, understanding how these different EC approaches work and how to apply them has developed a great deal over the past thirty years.

A discussion of some specific examples of particular evolutionary algorithms will clarify how the choices I just discussed can lead to different kinds of algorithms. Perhaps some traditional EAs offer the best kind of examples. They have very characteristic design choices, and they are easy to find in the literature. Therefore, a brief, high level discussion of two very different canonical classes of EAs is provided below: genetic algorithms and evolution strategies. After reading about these two types of algorithms, it should be clear that, while an ES and a GA may share some basic inspirational concept, as well as the same abstract algorithmic structure, they are very different algorithms, both in instantiation and in philosophy of application.

Genetic Algorithms

Arguably the most well-known class of EAs are genetic algorithms (GAs), pioneered by Holland (1992) and later popularized by Mitchell (1997, Goldberg (1989, De Jong (1975). GAs are chiefly characterized by their representation and selection method. Most GAs represent individuals using a binary encoding of some kind and select individuals using a stochastic method that biases selection by preferring more fit individuals over less fit ones by a degree proportional to their respective fitness. This so-called *fitness proportionate* selection tends to offer a relatively weak form of selection in which the effectiveness depends a great deal on the content of the population, as well as properties of the problem domain.

The binary nature of the representation in GAs strikes a clear difference between *genotype* (the genetic makeup of an individual) and *phenotype* (the expression of those genes), as well as the encoding/decoding transformations needed to navigate between the two. Nevertheless, the genotypic aspect of this representational choice calls to mind some rather traditional and obvious genetic operators, as well. Often GAs are characterized by mutation operators that either flip a randomly chosen bit in a binary string (e.g., 1-*bit mutations*) or, more commonly, consider each bit independently for flipping at some established probability level (so-called *bit-flip mutation*). Crossover is generally performed by randomly determining one (*1-point crossover* (Holland 1992)) or more (*n-point crossover* (De Jong 1975)) points in two binary strings and swapping the intervening pieces. Other forms of crossover operators include ones in which respective positions on the binary strings are considered for swapping independently (*uniform crossover* (Syswerda 1989) and *parameterized uniform crossover* (Spears and De Jong 1991)).

From a dynamics point of view, there are two typical GA methods: generational and steady-state. In the former, parents are selected (again, typically by proportionate selection) and used to generate an offspring population of the same size as the parent population, totally replacing the older generation with the new (Holland 1992). Preserving one or more of best members of the population is also not uncommon (generally referred to as *elitism*) (De Jong 1975). In a steady-state GA, parents are also typically selected using a fitness proportionate method, but offspring are generated one at a time and replace the worst member of the population (Rogers and Prügel-Bennett 1998; De Jong and Sarma 1992; Syswerda 1990; Whitley 1989). Generally this replacement is handled by removing the worst member after insertion, but such decisions can vary.

Using a genetic algorithm evidences a kind of philosophical choice on the part of an engineer employing such a method: it is “easier” to map to and from a more general representation than it is to design problem specific genetic operators. GAs are very portable algorithms, save for the genotype to phenotype mapping, and have often appeared to work moderately well on a large group of problems when very little is known about the search space of the problem in any representation. Its deficiencies center around the fact that the utility of the genetic operators is highly dependent on the mappings that must be done, and thus the success or failure of the algorithm is bound to such decisions.

Evolution Strategies

Another very powerful and very common class of EAs are so-called *evolution strategies* (ESs) (Schwefel 1995; Rechenberg 1973). Although ESs commonly employ a real-valued representation (that is, each individual can be seen as an array of real numbers) and operators appropriate for such a representation, perhaps their most defining characteristics are their distinctive dynamics. The simplest view of an ES is one that considers selection in two phases: an offspring population is produced from a parent population (often of different sizes), then a new parent population is produced from one or both of these populations. Parents contributing to the offspring pool are often selected uniformly at random from the potential parents, while survival is accomplished by truncation. That is, the new parent population consists of only the best of the competing individuals.

As the phrase "one or both of these populations" in the previous paragraph implies, ESs (like GAs) have two different types of dynamics (here called "strategies"): *plus strategy* and *comma strategy*. In a $(\mu+\lambda)$ ES, the λ offspring compete for survival directly with each other *and* the μ parents. Here the μ most fit surviving individuals are used to populate the next generation out of an aggregate population consisting of $\mu+\lambda$ individuals. Things are far less aggressive in a (μ,λ) ES, where parents have no hope of making the next generation. Truncation survival selection is performed only on the λ offspring, suggesting (of course) that $\lambda > \mu$.

Mutation is often the only genetic operator employed for an ES, though certainly crossover operators exist and are not infrequently applied. By far the most common mutation operator is one in which each gene is modified by some delta selected from a Gaussian distribution, the mean of which is 0, and the standard deviation of which is typically adapted as the run proceeds. The specifics of how this adaptation works is beyond the scope of this minimal description, so the reader is referred to other sources for this information (e.g., Beyer (2001, Bäck (1996, Schwefel (1995)). Recombinatory operators include something very similar to uniform crossover, as well as more representation specific operators such as *geometric crossover*, where the genes are treated as points in *n*-dimensional space and offspring are produced in the hypercube subspace defined by two such points (the parents) (Bäck 1996).

Evolution strategies have often proved to be quite good at problems that involve optimization of real-valued functions. Their use intimates a more focussed approach to solving particular kinds of problems, and suggests that the engineer understands the nature of the representation space to some degree (at least to the extent that a real-valued representation is more appropriate). However, its rigid representational assumptions and corresponding operators diminish its portability to some extent.

2.1.3 A Need for Something More

One aspect of analyses of EC, both theoretical and empirical, that has emerged is that EAs are by no means a panacea for complex problems. While a well-tuned and appropriately designed EA can often perform robustly on many problem domains, finding such an EA is often less than obvious. It is not uncommon for design engineers to easily construct an EA that performs decently on a problem, while finding it frustratingly difficult to design an EA that truly performs well. Moreover, the more complex the representation, the harder it is for one to gain intuition about the effects of operators and selection methods used in the algorithm.

Still worse, some problems seem to admit no real objective measure, and it can be very unclear how to apply a traditional, single-population EA. Engineers are faced with an important question: when one is faced with such problems, does one reject EC entirely, or does one return to nature to find a way of augmenting these algorithms to address these issues?

2.2 Overview of Coevolutionary Computation

Indeed, with varied success, nature-inspired heuristic EAs have been applied to many types of difficult problem domains, such as parameter optimization and machine learning. Both the successes and failures of EAs have lead to many enhancements and extensions to these systems. A very natural, and increasingly popular extension when problems domains are potentially complex, or when it is difficult or impossible to assess an objective fitness measure for the problem, is the class of so-called *coevolutionary algorithms* (CEAs). In such algorithms, fitness itself becomes a measurement of interacting individuals. This ostensibly allows the potential for evolving greater complexity by allowing pieces of a problem to evolve in tandem, as well as the potential for evolving solutions to problems in which such a subjective fitness may, in fact, be necessary (i.e., game playing strategies).

This section provides a brief overview of CEAs. I will keep things at a very high level, using the first part of the section merely to define coevolution since a much more specific discussion of the particular framework used for this dissertation will be introduced in the next chapter. After defining coevolution, I will offer high level discussions of cooperative versus competitive coevolution, the advantages coevolution may offer, and finally some of the challenges facing coevolution itself.

2.2.1 Defining Coevolution

Evolutionary biologist Price (1998) defines *coevolution* as “reciprocally induced evolutionary change between two or more species or populations.” While this definition is perhaps intuitively quite clear from the biological perspective, the term “coevolution”, as it is used in the evolutionary computation community, is far from a luminously or uniformly defined one. Researchers studying coevolution debate whether or not the term can be applied when there is only one population, for instance, versus when there are many. Some researchers suggest that the problem’s nature itself is imbued with whatever characteristics are required to consider an algorithm “coevolutionary”, and that some algorithms typically called CEAs, are not coevolution when applied to non-coevolutionary problems. Regardless, there is one common property about which most, if not all, coevolutionary computation (CoEC) researchers agree: individual fitness is *subjective* (Watson and Pollack 2001) in the sense that it is a function of its interactions with other individuals.

This realization does not clarify the distinction much, however. What precisely is the nature of the interaction? Do the interacting individuals have to be in different populations? Do they have to be current, or can they be plucked from the history of some extinct population? Are algorithms that apply selection methods that are inherently subjective (as, for instance, fitness proportionate selection is) “coevolutionary”? These are all reasonable questions, and there is very little beyond some semantic philosophical position that can be used to differentiate some groups from others in these ways. In such matters, it is best to be pragmatic, however, and suggest that distinctions can be made only insomuch as they help elucidate study. Therefore, I will not eliminate alternative versions of a definition for coevolution, but will instead clarify terminology based on such pragmatism. I will, however, provide much more detail about these issues in the next chapter.

The real trouble with understanding the term is exactly the source of the algorithm’s inspiration: biology. In biology, *all* evolution is coevolution by the above property, because individual fitness is a function of other individuals by the definition of evolution. In EC, however, traditional EAs use an artificial, typically objective, fitness measure—one that is very alien to the biological world. Thus, the distinction between objective and subjective fitness becomes necessary to us, merely by virtue of the way in which EAs are applied to problems. Since it is problem solving in which I am primarily interested, it makes sense to define terms accordingly, by their utility in engineering: by what and how they measure something. In order to build computational models of evolution that are meant to solve problems, one must measure individuals in a population, that is establish a measure of value for a given representation of a potential problem solution, or component of a problem solution. The following four definitions are provided to help the reader understand the different types of possible measures for individuals.

Definition 1. ***Objective measure*** **–** *A measurement of an individual is* ***objective*** *if the measure considers that individual independently from any other individuals, aside from scaling or normalization effects.*

Definition 2. ***Subjective measure*** **–** *A measurement of an individual is* ***subjective*** *if the measure is not objective.*

Definition 3. ***Internal measure*** **–** *A measurement of an individual is* ***internal*** *if the measure influences the course of evolution in some way.*

Definition 4. ***External measure*** **–** *A measurement of an individual is* ***external*** *if the measure cannot influence the course of evolution in any way.*

Understanding, at a high level, what is meant by the various kinds of measurements available to the individual helps one gain a better understanding of EAs and CEAs. For example, it is clear from above that the term "fitness" as it is applied by people in the EC community, is always an *internal measure*. External measures for traditional EAs might include tracking statistics such as mean fitness of a population, or best-so-far information. Indeed, a popular statistic reported is the external *best-ever* measure of a run (the most optimal result found during the life-time of the search).

Given the above definitions, it is tempting to define coevolution as follows:

Definition 5. ***Coevolutionary algorithm*** *– An EA that employs a subjective internal measure for fitness assessment.*

Since fitness proportionate selection merely normalizes the population's fitness value, I can eliminate such an operator as an instigator of ambiguity for denotational purposes; however, there are still other mechanisms that create ambiguity, as I will explain in Chapter 3. In an EA the simplest way to differentiate species (groups of non-interbreeding individuals) is by employing separate populations. Thus, a broader question that still stands is the following: can one call it "coevolution" if there is only one population? In the strict biological sense, one cannot be so general since employing multiple populations is explicitly part of the definition. However, historically, EC researchers have done so, recognizing that these algorithms share common properties with multi-population coevolutionary models (primarily game-theoretic properties) (Ficici and Pollack 2000c). Moreover, as has already been mentioned, the biological definition cannot be directly applied in any event, since the notion of fitness differs so dramatically between EC and biology. Given this fact, the historical precedent, and the overlap in dynamical characteristics, it makes sense to retain the term for single-population models that use subjective fitness, though this dissertation will focus on multi-population models almost exclusively, so this distinction is unimportant to the analysis I present. As for the remaining grey area, it is left up to the reader to consider whether augmentations to traditional EA methods constitute sufficient interdependence in fitness assessment to be considered "coevolutionary". It suffices to say that the line between a single-population CEA and an augmented traditional EA is a grey one.

To be clear, in the case of this dissertation, the term *coevolutionary algorithm* almost exclusively refers to an algorithm in which there are two or more populations, and in which individuals are awarded fitness values based on their interactions with individuals from the other population(s). In the rare situations when single-population CEAs are discussed, they will be so qualified.

2.2.2 Cooperative versus Competitive CEAs

If CEAs are distinguished from traditional EAs on the basis of their use of subjective fitness, that individuals are evaluated based on their interactions with other individuals, what is the nature of these interactions? The answer is: it depends. It is not hard to imagine algorithms in which individuals or populations compete with one another. For example, consider a predator-prey model in which individuals in one population represent some kind of device (e.g., a sorting network) and individuals in another population represent some kind of input for the device (e.g., a data set), and the object of the first population is to evolve increasingly better devices to handle the input, while the object of the second population is to evolve increasingly more difficult inputs for the devices. Such algorithms are generally referred to as *competitive* CEAs. Alternatively, it is equally straightforward to consider an algorithm where each population represents a piece of a larger problem, and it is the task of those populations to evolve increasingly more fit pieces for the larger, holistic problem. Such algorithms are generally referred to as *cooperative* CEAs (CCEAs).

Historically, competitive CEAs lead the way with the seminal Hillis (1991) paper on coevolving sorting networks and data sets in a predator-prey type relationship. Hillis evolves sorting networks by using an opposing population of coevolving data sets. In this case, an individual in one population, representing a potential sorting network, is awarded a fitness score based on how well it sorts an opponent data set from the other population, and individuals in the second population represent potential data sets whose fitness is based on how well they confuse opponent sorting networks.

In fact, most work in coevolutionary algorithms has been in the area of competitive coevolution. Most popularly competitive coevolution has been applied to game playing strategies (Rosin and Belew 1995; Rosin and Belew 1996; Rosin 1997; Pollack and Blair 1998). Additionally Angeline and Pollack (1993) demonstrate the effectiveness of competition for evolving better solutions by developing a concept of competitive fitness to provide a more robust training environment than independent fitness functions. Competition was also successfully harnessed by Schlierkamp-Voosen and Mühlenbein (1994) to facilitate strategy adaptation in their so-called breeder genetic algorithms. Competition has played a vital part in attempts to coevolve complex agent behaviors (Sims 1994; Luke, Hohn, Farris, Jackson, and Hendler 1998). Finally, competitive approaches have been applied to a variety of machine learning problems (Paredis 1994; Juillé and Pollak 1996; Mayer 1998).

Potter and De Jong (1994) opened the door to research on cooperative CEAs by developing a relatively general framework for such models and applying it, first, to static function optimization and later to neural network learning (Potter and De Jong 2000). In Potter's model, each population contains individuals representing a component of a larger solution, and evolution of these populations occurs almost independently, in tandem with one another, interacting only to obtain fitness. Such a process can be static, in the sense that the divisions for the separate components are decided *a priori* and never altered, or dynamically, in the sense that populations of components may be added or removed as the run progresses.

Moriarty and Miikkulainen (1997) take a different, somewhat more adaptive approach to cooperative coevolution of neural networks. In this case a parent population represents potential network *plans*, while an offspring population is used to acquire node information. Plans are evaluated based on how well they solve a problem with their collaborating nodes, and the nodes receive a share of this fitness. Thus a node is rewarded for participating more with successful plans, and thus receives fitness only indirectly.

Potter's methods have also been used or extended by other researchers. Eriksson and Olsson (1997) use a cooperative coevolutionary algorithm for inventory control optimization. Wiegand (1998) attempts to make the algorithm more adaptively allocate resources by allowing migrations of individuals from one population to another in a method similar to the Schlierkamp-Voosen and Mühlenbein (1994) competitive mechanisms.

The differences between these two algorithms are neither minor nor clear. Purely competitive CEAs can behave quite differently than purely cooperative ones, exhibiting different pathologies, as well as different advantages. However, once a particular algorithm and problem domain are dissected for analysis purposes, it becomes clear that there are often elements of both cooperation and competition in many CEAs. Indeed, when one considers single-population CEAs, it is difficult to discern the difference between competition as a result of selection within the population, and competition as a result of the relationships in the subjective fitness assessment.

2.2.3 Advantages of Coevolution

Intuitively, coevolution offers a great deal of promise as an heuristic algorithm in many domains where more traditional evolutionary methods are bound to fail. Whether this intuition is justified or not is the subject of much debate and, in part, is the impetus for the current work. This section will briefly consider three categories of problem domains that may benefit from coevolution and briefly discuss the issues involved with each of them: problems with large (infinite) Cartesian-product spaces, problems with no intrinsic objective measure, and problems with complex structures.

In all three cases, the hope of coevolution is to produce a dynamic typically referred to as an *arms race*. Informally, in an arms race increased performance is generated by each population making incremental improvements over the others in such a way that steady progress is produced. The idea is that the system is *driven* to better parts of the search space by these reciprocal improvements, each getting better and better over time. Consider again the predator-prey example: the prey evolve to run faster; then the predator population is forced into evolving behaviors that thwart this advantage; then the prey evolve better hearing to detect their nemeses, and again the predators must change. The hope is that, in the end, both populations have exceptional attributes.

Large (Infinite) Cartesian-Product Spaces

In a traditional single-objective optimization problem, solutions are argument values that produce the highest function value of all arguments. Of course such problems can have large domain spaces; however, for some problems the search space is particularly and exceptionally large, even infinite. When this is the case, it is reasonable that no optimization procedure can be expected to find the result in any reasonable time, and instead practitioners become interested in finding interesting sub-spaces of the total space.

Often this takes the form of some kind of Cartesian-product space, as was the case with the example of sorting network and data sets. The total space is quite large. In fact, if we had to search the space of possible networks that correctly sort every possible data set with a traditional EA, the result would obviously be less than useful. One potential solution is to select a specific, static subset of data sets that serve as useful teaching examples (Rosin 1997); however, the result will be an EA that is very well-suited for those particular examples, but not necessarily other data sets. Another potential solution is to use a random set of examples; however, since the example space is so vast in many cases, the result is often that the EA is unable to learn anything useful at all.

Coevolution offers something different: a problem solver that adapts both parts of the product space together. The hope is that the algorithm will focus on parts of the example space that are useful and interesting, learning sorting networks that serve these exemplary data sets very well.

Problems with No Intrinsic Objective Measure

Still more complicated are problems in which there is no intrinsic objective measure. Such is often the case with game-playing strategies for instance. In many instances, games have *intransitive* relationships that complicate objective measurement; strategy *a* can beat strategy *b*, *b* can beat *c*, and *c* can beat *a*. For example, in Rodenberry's Star Trek, James Kirk, a relatively poor chess player, was able to beat Spock, a relatively advanced player, simply because he confounded Spock's expectations by playing "illogically". Certainly it is not hard to imagine players that Spock could beat, even though Kirk never would be able to do so. Such is the nature of these intransitive relationships.

These intransitive relationships may comprise only minor portions of the strategy space, or may permeate the entire space. Especially in the latter case, is it the goal of a learner to develop a strategy that beats as many other strategies as is possible, or is it the goal to beat strategies that are considered quite good? Supposing the former, how would a traditional EA solve such a problem? Again, perhaps a suitable teaching set is used and the same brittle result is obtained, and again, perhaps a random opponent is used with the same poor result.

Without an intrinsic objective measure, coevolution offers something even more than co-adaptation: an implied answer to the question of "What is best"? Since (as I will discuss later in the dissertation) these algorithms are predisposed toward Nash equilibria, coevolution offers the opportunity to find strategies that are as non-dominated as is possible.

Complex structures

While traditional evolution may be fully applicable to static single-objective optimization problems of arbitrary complexity, the decompositional nature of coevolution (whether implicit or explicit) may afford CEAs with some advantages for dealing with problems that are complex, but highly structured. Assuming that the algorithm either endogenously or exogenously decomposes the problem in an appropriate way, it seems natural that a CEA (in particular a CCEA) could coevolve the various components independently more efficiently than could a traditional EA evolve the entire structure.

Indeed, this has been the primary motivating factor for cooperative coevolutionary approaches from the beginning, starting with early attempts to perform optimization tasks with explicit, static decompositions (Potter and De Jong 1994). Soon researchers began to explore controlled, but dynamic decompositions (Potter and De Jong 2000), and even more recently CCEA approaches have exploited this notion further, employing a kind of shaping technique called *complexification* (Stanley and Miikkulainen 2002; Stanley and Miikulainen 2002). Here the structure that is coevolved starts out simple and is gradually expanded to help direct the search space more hierarchically from simple representation spaces (where the search space is much smaller) to larger representation spaces (where the search space is constrained by the earlier parts of the search).

2.2.4 The pathologies of coevolution

Despite the idealistic hopes of the power of coevolution, applications of CEAs (both cooperative and competitive versions) often fail, or the algorithms turn out to be far more difficult to tune than are traditional EAs. The reasons for this lie partially in measurement problems caused by the use of subjective fitness and partially in the often particularly complicated dynamics of coevolutionary systems. These two difficulties together can lead to a system behaving incomprehensibly at times and in which progress measurement issues also make such behaviors difficult to diagnose.

There are a variety of fairly traditional pathological dynamical behaviors for coevolutionary algorithms; however, they tend to be poorly defined in the literature. I will describe the historical terms here at a high level, but later in the dissertation I will modify the terminology, as well as discuss the pathologies in more detail.

Perhaps the most common pathology is the so-called *loss of gradient* problem, in which one population comes to severely dominate the others, thus creating an impossible situation in which the other participants do not have enough information from which to learn (e.g., a small child, new to the game of chess, attempting to learn to play by playing a grand master at her best). Another common problem is *cyclic behavior*, where intransitivities in the reward system can allow one population to adapt slightly to gain an advantage over the others, then the others follow suit, only for the original population to change again, eventually, back to the original strategy. A similar, but subtly different pathology is that of *mediocre stability*, also referred to as “relativism” (Watson and Pollack 2001). Here stable limiting behaviors (either fixed-point or cyclic) are obtained, but do so at particularly suboptimal points in the space, from some external perspective. This is a kind of disconnection between what the engineers have in mind and what the system finds most natural. Finally, CEAs can have *focussing problems*, often producing brittle solutions because the coevolutionary search has driven players to over-specialize on their opponent’s weaknesses. Defining, diagnosing, and treating these problems has been at the forefront of CoEC research.

The measurement issue mentioned above has also occupied a great deal of attention of coevolutionary researchers, who have appropriated the biological term *Red Queen* to help understand this diagnostic problem. The difficulty is that since fitness is internal and subjective, it is impossible to determine whether these relative measures indicate progress (supposing one has an external notion of “progress”) or stagnation when the measurement values do not change much (or even the reverse in other cases). Without engaging some kind of external or objective measure, it is difficult to understand what the system is really doing. Perhaps an arms race is occurring, or perhaps the system has stagnated in some mediocre, but stable part of the space. Because of the relative measurements in the system, it is impossible to know which is the case. Here one should be clear about what is meant by the term *Red Queen effect*, since it is defined somewhat differently in different places (Watson and Pollack 2001; Pagie and Hogeweg 2000; Ficici and Pollack 1998; Cliff and Miller 1995; Dawkins and Krebs 1979). In fact, Red Queen dynamics are neither bad nor good or, more precisely, they *may* be bad or good, but it is impossible to tell which. A more precise definition follows.

Definition 6. ***Red Queen effect*** **-** *The diagnostic problem that occurs when populations seem to be changing, but the internal subjective measure shows no progress is occurring. The phenomena may describe stagnation or arms race. It is created by the fact that observing internal subjective fitness measurements provides no external information about the behavior of the system.*

2.3 Background Work in Analysis of CoEC

Until recently, analytical work in the field of coevolutionary computation was virtually non-existent. However, the past decade and a half has seen an explosion of introductory work in the area. This section reviews this work in four categories: component analysis, performance and problem measures, Markov and dynamical systems analysis, and asymptotic run time analysis.

2.3.1 Component Analysis

Component analysis of CEAs has primarily taken the form of empirical analysis of methods of interaction, effects of problem decomposition, and effects of genetic operators. Of these three categories, the first has been by far the most explored.

How an individual is paired up to evaluate fitness in a coevolutionary algorithm is not a small decision. Naively, practitioners might want to perform complete pairwise interactions with all possible combinations of individuals (what will later be termed *complete mixing*); however, this is obviously highly computationally inefficient. Several studies have focussed on understanding how these decisions should be made for certain kinds of problems. Perhaps the earliest such study is Angeline and Pollack (1993), where empirical evidence was given regarding the effects of different topologies of competitive tournaments. Here it was shown that for some kinds of problems, a simple single tournament matching was sufficient for establishing a good measure of an individual's qualitative value, while sometimes more complicated mechanisms were required, more tournaments per individual necessary. A later extension of this study done by Luke and Panait (2002a) examines two specific competitive fitness mechanisms, single-elimination tournament and *k*-random opponents, finding that single elimination indeed seems to do well when there is little to no noise in the fitness function; however, the *k*-random mechanism seems better in noisy situations.

Even in its introductory state, the original cooperative coevolutionary paper by Potter and De Jong (1994) offers some minimal empirical evidence that non-linearities in problem components split across populations may require more sophisticated choices in how one makes choices of interactions for fitness purposes. A far more comprehensive study, involving many different methods for selecting "Partners" was made by Bull (1997), coming to a similar conclusion: non-linearities between populations may create the need for more sophisticated methods of interaction.

This early focus on the absence or existence of non-linear relationships between parts of problems split between populations (so-called *cross-population epistasis*) established an almost myopic focus on this property. The natural question of how one should decompose the problem, and how that affects choices in the algorithm seemed to intermingle with the question of method of interaction. Bull extended his study to include some formalism (Bull 2001) using Kauffman's NKC (Kauffman 1991) as a means of categorizing problems by their degree of cross-population epistasis. A more surgical empirical analysis, exploring various aspects of the decisions factors involved in choosing a method of interaction, was provided by Wiegand, Liles, and De Jong (2001), indicating that the issues at stake may not necessarily be merely the property of cross-population epistasis at all. This study was followed by Wiegand, Liles, and De Jong (2002a), where it becomes clear that there are different *types* of epistasis that can have different impacts on this decision. Both of these studies form the basis a some of the research discussed in chapter 3.

In his 1997 dissertation, Potter (1997) conducts a small empirical study to attempt to understand the effects of static problem decomposition by analyzing a string matching problem, in which the degree of overlap of the match is controlled for the study. This study suggests very little save that, for some problems, even with strong overlap his general CCEA framework can still work reasonably well. A year later, Bull published a companion article to the above cited article on “partnership”, where he showed that the effects of mutation on CEAs can also be quite sensitive to problem properties (Bull 1998).

2.3.2 Performance and Problem Measures

Perhaps the first work attempting to establish a method for external (though still subjective) measurement of CEAs was Cliff and Miller (1995), where a method using the history of the run itself is employed to help track and diagnose dynamics in CEAs. An information-theoretic approach was used by Ficici and Pollack (1998) in order to provide a measure that is both external and objective for the purposes of such diagnostics. This method helped them determine how, in certain settings, arms race behaviors might be established. More recently, Stanley and Miikkulainen (2002) developed an external, subjective measure based on dominance that can be useful for comparing two different CEAs, but may provide less useful information about the types of dynamics exhibited in each.

In addition, several empirical studies also attempted to compare search dynamics between CEAs and traditional EAs using problems with implicit external measures (Pagie and Mitchell 2001; Juillé and Pollack 1998), showing that arms race dynamics can often be quite tricky to establish, and that population structure choices (such as spatial embedding) may be of some help. Watson and Pollack (2001) establishes a precedent for using a very simple problem medium in order to understand evolutionary dynamics, which was followed by a much more formal study by Bucci and Pollack (2003) illustrating that many types of coevolutionary problems can be reduced to this simple medium, and that the problem of focussing may be more common than many CoEC researchers believe.

The above work was based on an order-theoretic approach for helping to define problems that are in some sense intrinsically coevolutionary (Bucci and Pollack 2002). Luke and Wiegand (2002), another formal study, also shows that the order relationships in problem’s reward system can suggest characteristics about the dynamics, indicating that in certain cases coevolutionary dynamics in single-population CEAs can be equivalent to traditional EA dynamics. In addition to these works, Olsson (2001) provides an analysis that suggests that asymmetries in coevolutionary problems can have profound effects on their dynamical behaviors.

Not only has analysis been done to help understand and diagnose dynamical behaviors in CEAs, but some limited amount of effort has been employed to try to make use of that knowledge in improving the search, such as the afore mentioned Ficici and Pollack (1998) paper. Additionally, Rosin and Belew (1997, Rosin and Belew (1995) suggest mechanisms such as “hall of fame”, “competitive shared fitness”, and “shared sampling”, harnessed through a rough form of static equilibria analysis, to help improve coevolutionary search by providing a higher probability of finding adequate “teaching sets”. In addition to this approach, Ficici and Pollack (2001) considers the overlap between multi-objective problem characteristics and coevolutionary problem characteristics to use the notion of Pareto optimality to help improve coevolutionary search.

2.3.3 Markov and Dynamical Systems Analysis

Dynamical systems approaches constitute perhaps the most promising category of theoretical analysis of coevolutionary algorithms. These models have primarily become popular as a result of two, related influences. The first source of inspiration comes from the fields of biology and economics, where the notions of evolutionary game theory (EGT) (Hofbauer and Sigmund 1998; Weibull 1992; Maynard-Smith 1982) have helped illustrate dynamical, “evolutive” (steady-state, in the dynamical systems sense) (Osborne and Rubinstein 1998) behaviors of systems that are very similar to coevolutionary systems. The second source is from within the EC community itself, where dynamical systems models have become increasingly more popular tools for understanding the dynamical properties of EAs (Reeves and Rowe 2002; Vose 1999).

The early EGT model model introduced by Maynard-Smith was originally applied, with little success, to attempt to predict behaviors in an EA by Fogel, Andrews, and Fogel (1998, Fogel, Fogel, and Andrews (1995, Fogel and Fogel (1995). However, these studies were somewhat remiss with how they matched the assumptions of the model with those of the algorithms being modeled. Ficici and Pollack (2000a), in an effort to re-apply this method, was able to match behaviors of real algorithms to the model predictions much better than these earlier works. Moreover, Ficici and Pollack (2000c) can perhaps be credited with offering the first real promise of using EGT for understanding dynamical behaviors in CEAs, from which it has been learned that certain types of traditional selection methods may be pathological in certain single population EAs (Ficici and Pollack 2000b). This EGT model was then turned towards cooperative CEAs (Wiegand, Liles, and De Jong 2002b; Wiegand, Liles, and De Jong 2002a), where it begins to become clear that the dynamical properties of CCEAs may not be as simple as previously believed. Extensions and refinements to this work constitute a large portion of this dissertation.

In very recent years, use of these dynamical systems approaches for coevolution has begun to become quite prevalent. Quite independently of the EGT work, efforts by Schmitt (2001) apply Markov modeling methods to EAs in which the fitness of individuals is population dependent, including competing agents under the conditional that an overall dominant strategy exists. Again, independent of the above work, Subbu and Sanderson (2000) uses a dynamical systems model with an altogether different framework in order to analyze distributed cooperative coevolutionary algorithms. This work, while being heavily entrenched in assumptions regarding Gaussian mutation operators and Gaussian-type problem properties, demonstrates surprisingly powerful results regarding the convergence properties of such algorithms. Also, recent work by Liekens, Eikelder, and Hilbers (2003) has applied Markov modeling techniques as a means of comparing CEAs, and in fact performs such an analysis as a means to understand the benefits of diploidy in coevolution. Additionally, Schmitt (2003) has further developed his Markov model for EAs to include certain kinds of CEAs.

2.3.4 Asymptotic Run Time Analysis

Though still in the very early stages, some recent research incorporates more traditional computer science asymptotic analytical methods, such as those applied toward other types of randomized algorithms (Motwani and Raghavan 2000). Jansen and Wiegand (2003b) presents the run time performance of a very simple cooperative coevolutionary algorithm on several separable problem classes in terms of expected time before the global optimum is reached first. This research again casts doubt on the older belief that cooperative coevolution gains advantage when there is little or no non-linear relationships between pieces of the problem represented by different populations. I will cover such analyses in greater detail in Chapter 4. Further analysis of this sort was applied by Jansen and Wiegand (2003c) to illustrate differences between two different implementations of a simple CCEA, one parallel and one sequential in terms of how the current populations are updated. This study suggests that the differences and similarities between such implementations are, perhaps, counter intuitive.

Chapter 3

Coevolutionary Architectures

In the previous chapter, I introduced a tentative definition of a coevolutionary algorithm: an evolutionary algorithm in which the fitness of an individual is a subjective function of its interactions with other individuals. An exact definition of just precisely what is, or is not, coevolution will not be offered; however, in order to study something, one must certainly clarify the object of study itself. This chapter seeks to do exactly this: lay the groundwork for understanding the specific class of coevolutionary algorithms on which this dissertation focusses.

The chapter begins this clarification by first offering a categorical description of the properties of CEAs. This has the double advantage of informing the reader of choices available to CoEC designers, while also providing a context for understanding the choices made for this particular set of studies. With this beginning, the second section is able to position the *Cooperative Coevolutionary Algorithm* (CCEA) framework under study here within the larger hierarchy of CEAs in general. Since the point of my fundamental question centers around the notion of static optimization, the third section seeks to clarify this problem domain for the CCEA. The result of this chapter should be a clear understanding of the algorithm I am studying and the problem to which some seek to apply it.

3.1 Categorizing Coevolutionary Algorithms

Just as with more traditional EAs, there are many design choices available to engineers employing coevolutionary algorithms. Different choices lead to different CEAs with different properties; thus, it is in a designer's best interest to understand these choices. Some of the choices have already been touched upon in the previous chapter, but it is useful to provide a systematic, hierarchical elicitation of them in greater detail. Indeed, this section offers a categorical ontology of coevolutionary algorithms, based on the choices available to algorithm designers. The descriptions of these categories follow Figure 3.1 on page 23, which provides a roadmap for these descriptions.

A categorization such as this is necessary if one is to take a systematic approach to understanding coevolution. Any given analysis of a class of coevolutionary algorithms can be made more clear (and specific) by directly addressing which subset of properties are explored in that analysis. Knowing a reasonably thorough ontological breakdown of properties helps researchers and practitioners understand what the generalities, as well as what the limitations, of such an analysis must be. Moreover, once a broad range of properties is more obvious in some holistic sense (as is hopefully the case with the hierarchy on page 23), more focussed and sensible analytical processes are possible by establishing a context. Additionally, a better understanding of what *is not* covered in prevailing research is illumined in this way.

As we will soon see, there is a large amount of semantic ambiguity surrounding many coevolutionary algorithms that make them difficult to classify. This difficulty only increases the need for some kind of categorization. By describing the various choices that go into defining a coevolutionary algorithm, the places where these ambiguities are the highest will be made more explicit.

As you can see from the figure on page 23, there are at least two high level branches to choices about coevolutionary algorithms: choices centering around how one evaluates fitness in a CEA and choices centering around how one represents the problem for the algorithm to solve. The following subsections offer short descriptions of each of these, and their subcategories, in turn.

3.1.1 Evaluation

Since CEAs are defined by the fact that their internal measure, or evaluated fitness, is a subjective measure, a logical breakdown of defining properties for CEAs should include evaluation as a high level category. Indeed, there are at least four different properties of evaluation important to coevolution: payoff quality, methods of fitness assignment, methods of interaction, and update timing. Though there may be other important properties of CEA evaluation, these four must certainly be the most significant, and within these categories nearly all existing CoEC approaches can be described.

Payoff Quality

The first broad category of coevolutionary algorithm evaluation surrounds the character of the payoffs awarded during interaction of individuals. This is essentially a tacit reference to the game-theoretic nature of coevolution that I will discuss later in the thesis. The basic question is: if one is evaluating interactions, how are these interactions best characterized? There are at least three payoff types in CEAs worth considering: competitive fitness, cooperative fitness, and non–competitive fitness. The reader should be careful to note that these are traditional, qualitative classifications that leave a lot of grey area to be disambiguated. In fact, there is much disagreement about the scope and relevance of these grey areas within the field of CoEC. I will return to this question of ambiguity below, but first let's say a few words about the categories themselves.

In *competitive coevolution*, individuals are rewarded when their opponents do poorly, and they are punished when their opponents do well. In other words, their fitness values depend on *competitions*, and the individuals participating in these competitions are considered *competitors* or *opponents*.

Alternatively, in *cooperative coevolution* the fitness of an individual is based on how well it cooperates with the other individuals with whom it interacts. Individuals succeed when they are partnered well, when the resulting team performs well. Here I term the interaction itself a *collaboration*, and use the term *collaborator* to mean those individuals with which the interaction is made.

Though the definitions offered for distinguishing between cooperative and competitive coevolutionary methods remain somewhat vague, one thing that is clear is that one can imagine payoff situations where sometimes participants are mutually rewarded, but sometimes their purposes are at odds. Such situations are often referred to as *non–competitive*. This thesis will ignore non–competitive approaches since they are (so far) of little interest to those who study CEAs at this time. Nevertheless, it should be clear that it may well be the largest of the three groups.

Returning to my point about the ambiguity of these definitions, the question of how to clearly define these groups of payoff quality is a very messy and complicated semantic problem. There's little doubt that more precise, mathematically rigorous definitions of the distinction between cooperation and competition may well be helpful. Indeed, such distinctions may be plausibly obtained using the notational tools from game theory (Rapoport 1970); however, the differences of opinions among CoEC researchers currently regarding subtleties of what is meant by these terms makes any unilateral attempt to draw a hard defining line between cooperative and competitive coevolution unwise. This dissertation, therefore, will not attempt to do so. Instead, as will be seen later in this chapter, the studies presented here will restrict their attention to a subset of such models that are undeniably cooperative.

Perhaps of more fundamental interest than terminological distinctions, may be whether or not any particular characterization of this sort has any meaning in terms of identifying potential long term dynamics in a coevolutionary system. This remains an open question, though the connection between payoff quality and long term coevolutionary behaviors is widely believed to exist (Ficici and Pollack 2000c).

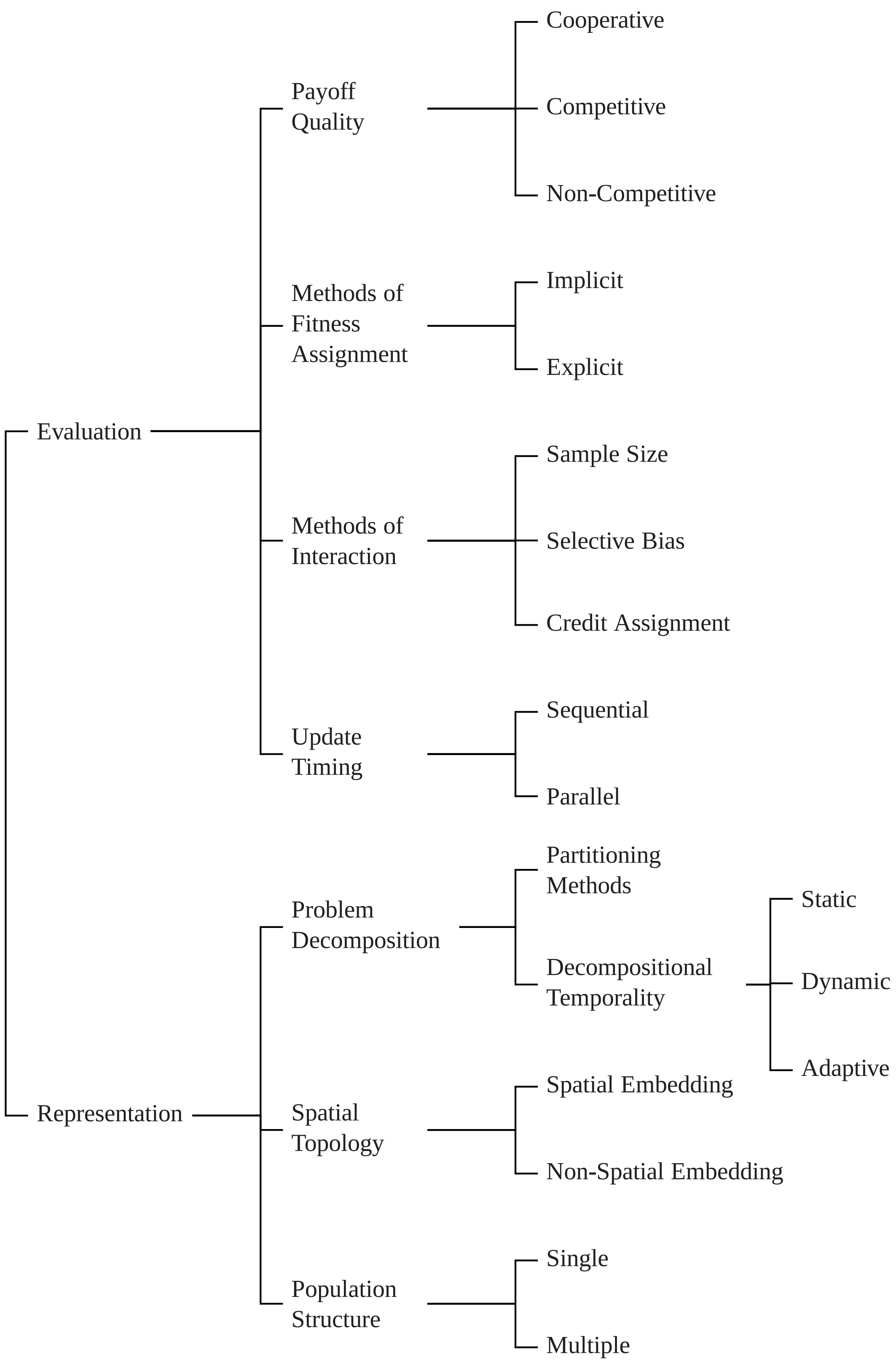


Figure 3.1: Hierarchical categorization of CEA properties.

Methods of Fitness Assignment

Conceptually there seems to be a clear difference between *calculating* a fitness score, and *using* such a value. A given EA may calculate fitness using some well–defined objective function, but that value may be used in a variety of ways either to influence the dynamics of the evolutionary search or to provide an external measure. Indeed, Chapter 2 discusses a variety of notions of measurement, some of which are applied directly towards search (so-called “internal measures”) and some of are not (so-called “external measures”).

Regardless of the specific issue of coevolution, internal measures can obviously be used in a variety of ways to influence the evolutionary search, independent of how they are used to finally report solutions to the problem. As an example, consider the case of applying parsimony measures in rule–based systems. In such cases evolving individuals are often complete rule-sets and penalties are dispensed to rule-sets that are too large (Luke and Panait 2002b). In these situations, one can make a distinction between *raw* fitness and fitness under some length bias pressure (for example). Although the length biased fitness is used during evolution, it is the raw/computed form of the fitness that has meaning to us from the perspective of solving a problem. An even more poignant example is proportional selection. In this case the *objective* fitness value calculated is taken to produce a *subjective* value, which is then used for stochastic selection—and is very clearly *not* coevolution.

However, if one’s focus is on what impact fitness has on the dynamic of the evolutionary search, it is primarily how it is *used* that is of interest, and less so how it is calculated. Thus, there are a variety of ways fitness can be used in both implicit and explicit ways that create the kind of subjectivity—a subjectivity based on interacting individuals one might feel more comfortable calling coevolution.

Examining the group of CEAs that use implicitly subjective fitness measures brings to one’s attention yet another source of ambiguity. Indeed, the line dividing what is *considered* a coevolutionary algorithm by the EC community at large and what is *considered* merely as some augmented traditional EA is a thin and fuzzy one. There are a large number of mechanisms introducing forms of implicit subjective internal fitness to traditional evolutionary algorithms that tend to be viewed at best peripherally as coevolutionary algorithms, but often not as CEAs at all.

Examples of this include methods for augmenting a GA to allow it to find *multiple* peaks in the fitness landscape frequently using a technique called *fitness sharing* (Goldberg and Richardson 1988; Spears 1994). In fitness sharing, the fitness of an individual is adjusted in some way based on the individuals *near* it in some way. Goldberg and Richardson (1988) uses a distance measure in order to reward diversity and punish phenotypic similarity, while Spears (1994) uses a set of *tag bits* in the chromosome in order to adjust fitness (as well as restrict mating) based on partial genotypic matches. Such algorithms tend not to be considered coevolutionary algorithms, but if one accepts the above coevolution definition, and by this include single population approaches presented by many evolutionary computation researchers (Angeline and Pollack 1993; Luke, Hohn, Farris, Jackson, and Hendler 1998), it is difficult to understand how methods that employ notions of fitness sharing can be excluded from the category of coevolution. Indeed, the main point of these methods in many cases is to help speciate the population (that is, separate the population into distinct, non-interbreeding species). The interaction that is happening, however, is implicit and quite subtle.

Though our working definition of coevolution is more or less clear in this respect, it is also clear that it cannot be taken too stringently. Applying the logic from the above paragraph can lead to strange inclusions into the coevolutionary camp. *Crowding*, for instance employs a method of diversity measure for the purposes of affecting replacement in a steady state GA (De Jong 1975). One might be tempted, at first blush, to say that this is not coevolution because the distance measures in this method are applied after fitness is calculated. This observation, however, is somewhat semantic since the dynamics of the algorithm are clearly affected by this interaction. Moreover, taking the above logic to the extreme, it might be argued that a simple GA using proportional fitness meets this vague definition of coevolution in the sense that selection is performed not on an absolute, objective measure, but instead on a relative measure that is dependent on other individuals in the current population. However, most EC researchers would not consider either EAs that employ crowding or canonical GAs using fitness proportional selection to be coevolutionary in nature, presumably because these effects are not really (typically) anything more than re-scaling. One might consider all such algorithms co-evolutionary, though, but if they are, the subjectivity of the fitness assessment is again implicit and even more subtle than what I've previously discussed.

Typically coevolutionary methods are very explicit about how their interactions occur, usually in a way that leaves no doubt about the interrelatedness of the fitness measures. When fitness is explicit, it is clearly stipulated by the algorithm designer, prior to running the algorithm. This is the case with most competitive approaches such as Rosin (1997), as well as many cooperative approaches such as Potter (1997). Such approaches are clearly forms of coevolution by almost any definition, though there are still a variety of differences between different types of CEAs that use explicit methods for subjective, internal fitness. CEAs using explicit fitness measures constitute the bulk of emphasis in the study of coevolutionary algorithms.

As a side note, it is interesting to observe that in multiple population approaches it can be true that fitness assignment in one population can be implicit, while in another it is explicit. This is the case in the already cited studies by Stanley and Miikkulainen (2002, Moriarty and Miikkulainen (1997), where individuals in subordinate populations receive fitness indirectly via their contributions to the performance of individuals in the master populations.

Methods of Interaction

The question of how a practitioner determines collaborators or competitors may very well be among the most important design decisions for the successful application of CEAs (Wiegand, Liles, and De Jong 2001). The most obvious (and computationally expensive) method to evaluate an individual in a coevolutionary setting is for an individual to interact with all possible collaborators or competitors. In the case of binary interactions, this is sometimes called *complete pairwise* interaction, or, more generally, in game theory it is often referred to as *complete mixing*. An alternative extreme is for an individual to be involved in only a single interaction. Such a choice leaves open the obvious question of how to pick the collaborator or competitor. Of course, between these two extremes is a whole host of possibilities that involve some subset of interactions. Again, collaborators/competitors for such interactions may be chosen in a variety of ways from uniformly random, to fitness biased methods.

Angeline and Pollack (1993) provide a reasonable summary for a variety of methods of choosing competitors, and most of that discussion is equally applicable to a cooperative context, as well. However, it primarily consists of the enumeration of several methods and a discussion about the relative merits of these specific strategies. Bull (2001, Bull (1997) also enumerate several methods for choosing “partners” (collaborators or competitors), but again the discussions center around the particular strategies chosen for these investigations.

While these efforts are helpful for specific cases, perhaps it is more useful here to define some basic attributes of this choice, suggesting a wide range of possible strategies (many of which may not yet have been conceived). There are mainly three such attributes: the *sample size*, *selective bias*, and *credit assignment* for potential interactions during fitness assessment. I describe these briefly below.

interaction sample size The number of collaborators/competitors from each population to use for a given fitness evaluation.

interaction selective bias The degree of bias of choosing a collaborator/competitor.

interaction credit assignment The method of credit assignment of a single fitness value from multiple interaction-driven objective function results.

Since how one makes such choices may be of high import to practitioners, much of the empirical research conducted by this thesis in Chapter 4 will center around determining how such choices affect the performance of existing coevolutionary algorithms. There may be other properties of this decision, but none so salient as these three.

Update Timing

Introducing multiple populations into an EA brings to the table some necessary clarifications about terminology. For example, some clarity with respect to the word *generation* is required. In a traditional generational EA, this typically refers one complete pass through the parent selection, offspring generation, evaluation, and survival selection (steps 4.1–4.5 in Algorithm 1 on page 8). I will retain this term in exactly this way for coevolution. That is, a *generation* in a CCEA is the time it takes to move through these steps for a single population. To refer to the complete cycle of processing all populations, I use the term *round*[1].

Next, do collaborators come from the current state of populations, or from populations from the previous round? Really this is a question of the timing of when populations are processed and updated. The timing of such things can be done in a variety of ways; I focus on two.

I use the term *sequential* to refer to a coevolutionary algorithm that processes each population in a sequential order, choosing collaborators from the current state of the other populations. These populations can change during the *round*, of course. As a result, populations processed and updated earlier in the round can affect the processing of populations processed later.

An alternate approach is one in which the individual populations are processed in *parallel*, synchronized only at the end of each round. In this approach, collaborators are selected from the *previous generation* from each population. A clearer description of the differences between these two approaches can be obtained in Jansen and Wiegand (2003c).

3.1.2 Representation

There are three particularly important groups of representation-oriented decisions that design engineers must make when implementing CEAs: how one handles problem decomposition, what is the spatial topology, and how one structures the population(s). Among these three categories, nearly all CEAs can be described one way or the other, though not all CEAs rely as heavily on some of these properties as do others.

Problem Decomposition

For cooperative coevolution in particular, it is often necessary to decide how a larger problem will be decomposed into smaller components. There are many ways one may perform this task; such methods fall into a category here called *partitioning methods*. Chapter 4 will explore the issue of partitioning methods and their effects more carefully.

The next question is *when* this decomposition occurs. Just like any parameter adjustment, the property of *decompositional temporality* may be *static*, decided once and for all *a priori* by the design engineer; *dynamic*, adjusted during run time using a rule imposed by the design engineer *a priori*; and *adaptive*, adjusted during run time by the system itself. The idea behind this distinction is similar to more traditional EA choices with respect to how parameters are altered (or not) at run time (Eiben and Michalewicz 1998; Angeline 1995).

The challenge of how and when to decompose a problem for the purposes of coevolving the individual components is one of the most important questions facing CoEC researchers. In fact, the nature of these decisions on the part of design engineers poses unique questions regarding the appropriateness of decomposition in terms of the true structure of the problem at hand. Since these may be very appropriate, or distinctly inappropriate, as will be seen later in the dissertation, it is important to make informed choices. Due to the difference between a

[1] I prefer the *generation/round* terminology to that offered by Potter (1997): *generation* versus *ecosystem generation*.

decomposition made by the representation and the natural decomposition of the problem, I use the term *component* to mean a part of a representation, given by the algorithm, while I use the term *piece* to mean portions of the problem itself.

Spatial Topology

Fine-grained, distributed representations of populations are important categories of traditional EAs (Sarma and De Jong 1996), but they are also quite important in coevolutionary algorithms, since they beg the additional question of how multiple populations relate to one another topologically (Pagie and Hogeweg 2000). Indeed, some early research has suggested that spatially-embedded CEAs might outperform non-spatial methods on some problems, in certain confined circumstances.

Of course, the bulk of CEAs use no such spatial embedding, but a modest subset of application oriented work now involves these fine-grained approaches. The research on such models, as well as its apparent potential advantage, suggests that this category should appear in our ontology. Design choices for fine-grained CEAs are complicated, but are for the most part outside the scope of this work. This dissertation distinguishes only between models that are *spatially embedded* and those that are *non-spatially embedded*. It should be noted, however, that topological relationships in spatially embedded models are inherently connected to methods of interaction in terms of evaluation.

Population Structure

There are many ways for a design engineer to break up a coevolutionary algorithm; however, the primary distinction made here is whether *multiple populations* or a *single population* is employed by a given CEA. These two mark off very different areas of coevolutionary research, so this division constitutes a highly useful distinction.

For our purposes, let us assume that in single population approaches individuals interact with other individuals within the same population for obtaining fitness, while in multiple population approaches individuals interact only with individuals from *other* populations to obtain fitness. While it is possible to consider algorithms in which there are interactions between individuals in the same population *and* other populations, it is relatively rare and unhelpful to consider such complications. Indeed, as it turns out, there are apparently some clear distinctions between single and multiple population approaches in terms of dynamics (Ficici and Pollack 2000c; Hofbauer and Sigmund 1998).

3.2 A Cooperative Coevolutionary Algorithm

In the previous section, I offered a categorical description of the design choices available to those wishing to construct coevolutionary algorithms. Using this hierarchy, nearly any conceivable coevolutionary algorithm can be described, even those that fall into the grey areas of what is considered coevolution. However, to say something specific, to answer direct and pointed questions, I must restrict our attention to a subset of CEAs that are most salient for the purposes of the study at hand.

In my case, every attempt has been made to keep the class of CEAs under study as simple as is possible. The ultimate goal is to preserve the coevolutionary nature of the algorithm, while providing an algorithm that gives rise to interesting and useful questions. Therefore, some of the choices available to us will be restricted in the interest of simplicity and focus, while others will be varied in order to answer practical questions about the class of algorithms considered. To help clarify this more specifically, I will divide my discussion into three parts. In the first two parts, I describe properties of the algorithm relating to evaluation and representation, respectively. In the final part, I will offer a more detailed description of the algorithms' framework.

3.2.1 Evaluation

While there are certainly interesting questions relating to both cooperative and competitive coevolution, cooperative coevolution may meet the "simple" criteria somewhat more directly. The lack of a clear external and objective measure for most competitive methods, in combination with the semantic difficulty defining optimality, makes progress measurement a problem. This creates challenges for study, since often this measurement problem must be solved before any more analysis can be accomplished. Moreover, existing research suggests that the dynamical properties of competitive systems may very well be more challenging to understand than those of cooperative systems. Since part of my goal is one of simplification, the algorithms studied in this dissertation will be explicitly cooperative, in the sense that they meet the qualitative definition of cooperative coevolution payoff quality and explicit fitness assignment offered in the previous section.

However, since one of the key decisions that face cooperative coevolutionary algorithm designers surrounds the method of interaction, this property will remain variable for our investigations. Indeed, the next chapter will explore the nature of this choice, and its effects in detail.

This issue of what kind of effects the timing of processing and updating populations has on the CCEA is a complex one. My analysis here does not include a discussion of the differences, though related work is cited that does (see Jansen and Wiegand (2003c), for example); however, I will concentrate primarily on sequential CCEAs in Chapter 4 and on parallel CCEAs in Chapter 5.

Before moving on to discuss representation, it is necessary to say a few words about the ambiguity of the term "cooperative" raised in the previous section. In the next section I will describe the problem domains to which this dissertation restricts its attention, and the definitiveness of the cooperative nature of the algorithms on such problems will be more clear. Despite the general ambiguity in this term, the most obvious and direct static single-objective optimization tasks suggest a cooperative approach. Since the notion of optimization is at the heart of the fundamental question of the dissertation, it seems to be the most appropriate decision under these circumstances.

3.2.2 Representation

For reasons of simplicity, the algorithms in this work use only *a priori* static decompositions, though the partitioning method will be the subject of some of the research in the next chapter since it relates strongly to methods of interaction. I primarily will focus on algorithms that are non-spatially embedded; however, in the final chapter a particular spatial model is used to make a parting point about establishing and maintaining evolutionary balance during the run. The primary research investigations of this thesis focusses entirely on non-spatial models, though.

Single population models can be quite messy and complicated to describe and understand. Moreover, it is unclear whether cooperative algorithms are even possible in such a setting, given that it is the nature of individuals in a population to compete for selection, regardless of the nature of their interactions. To avoid this complexity, as well as to place this research safely in the "coevolution" camp, I employ only multiple population approaches in which the fitness of an individual from one population is the result of collaborations with individuals from each of the other populations.

3.2.3 A General Cooperative Coevolution Framework

The discussion above defines a rather large class of potential CEAs. Unless otherwise stated, from this point forward the term *cooperative coevolutionary algorithm* (CCEA) will to refer this particular class of multi-population, cooperative algorithms using explicit fitness assignment methods and static problem decompositions.

When applying a CCEA to a particular problem, typically one decomposes the problem into components and assigns each component to a population. Save for evaluation, each population is evolved more or less independently of one another. since any given individual from a particular population represents only a component of a potential solution to the problem, collaborators are selected from the other populations to represent the remaining components. The individual is combined with its collaborators to form a complete solution and the objective function is evaluated. If there is more than one set of collaborators (depending on the collaborator sample size) for a given individual, the objective function may be evaluated many times, but a single score is somehow attributed to the individual for fitness (depending on the collaborator credit assignment method). It should be noted that the populations of a CCEA may or may not be homogeneous with respect to the representation used for the encoding of individuals in them, or the underlying EAs being used to evolve a particular component. For the sake of simplicity, all the studies in this dissertation will assume a more or less consistent representation across populations. Figure 3.2 on page 30 illustrates this idea pictorially.

This class of CCEAs is in no way new. Indeed, Potter (1997) describes a generalized framework for the CCEA, one that has been applied in a variety of ways. To be more specific, the traditional EA code shown in Figure 1 on page 8 is extended, describing the abstract *sequential* CCEA in the pseudo-code in Figure 2 below.

Algorithm 2 (Abstract Sequential Cooperative Coevolutionary Algorithm).

```
1. for population $p_s \in P$, all populations
   1.1 Initialize population $p_s$
2. for population $p_s \in P$, all populations
   2.1 Evaluate population $p_s$
3. $t := 0$
4. do
   4.1 for population $p_s \in P$, all populations
       4.1.1 Select parents from population $p_s$
       4.1.2 Generate offspring from parents
       4.1.3 Select collaborators from $P$
       4.1.4 Evaluate offspring with collaborators
       4.1.5 Select survivors for new population $P_s$
   4.2 $t := t + 1$
until Terminating criteria is met.
```

3.3 Optimizing with The CCEA

In the previous section, I detailed the specific class of algorithms I will be analyzing. Several things remain to be clarified, however. For one thing, there is again the question of to which class of problem domains to restrict attention. The answer to this question arises from both the need to meet the fundamental question (Do CCEAs optimize?), as well as meet the condition of being as simple as is possible to demonstrate useful things about coevolution. Perhaps the simplest possible domain one might consider for the CCEA is the that of single-objective, static function optimization.

Such problems have at least two properties of interest here. First, they suggest some direct and obvious methods of decomposition. Second, given fairly straightforward decompositions, they have a game-theoretic property that is important to the analysis: symmetry. Before turning to these issues, I will clarify what is meant by *single-objective, static function optimization*. I focus on maximization. The results and observations are equally applicable to minimization.

Definition 7. *Let there be some function, F, over domain D, $F : D \to \mathbb{R}$. A* single-objective, static maximization problem *is a problem in which the goal is to find $d^* \in D$ such that $d^* = argmax_D F$.*

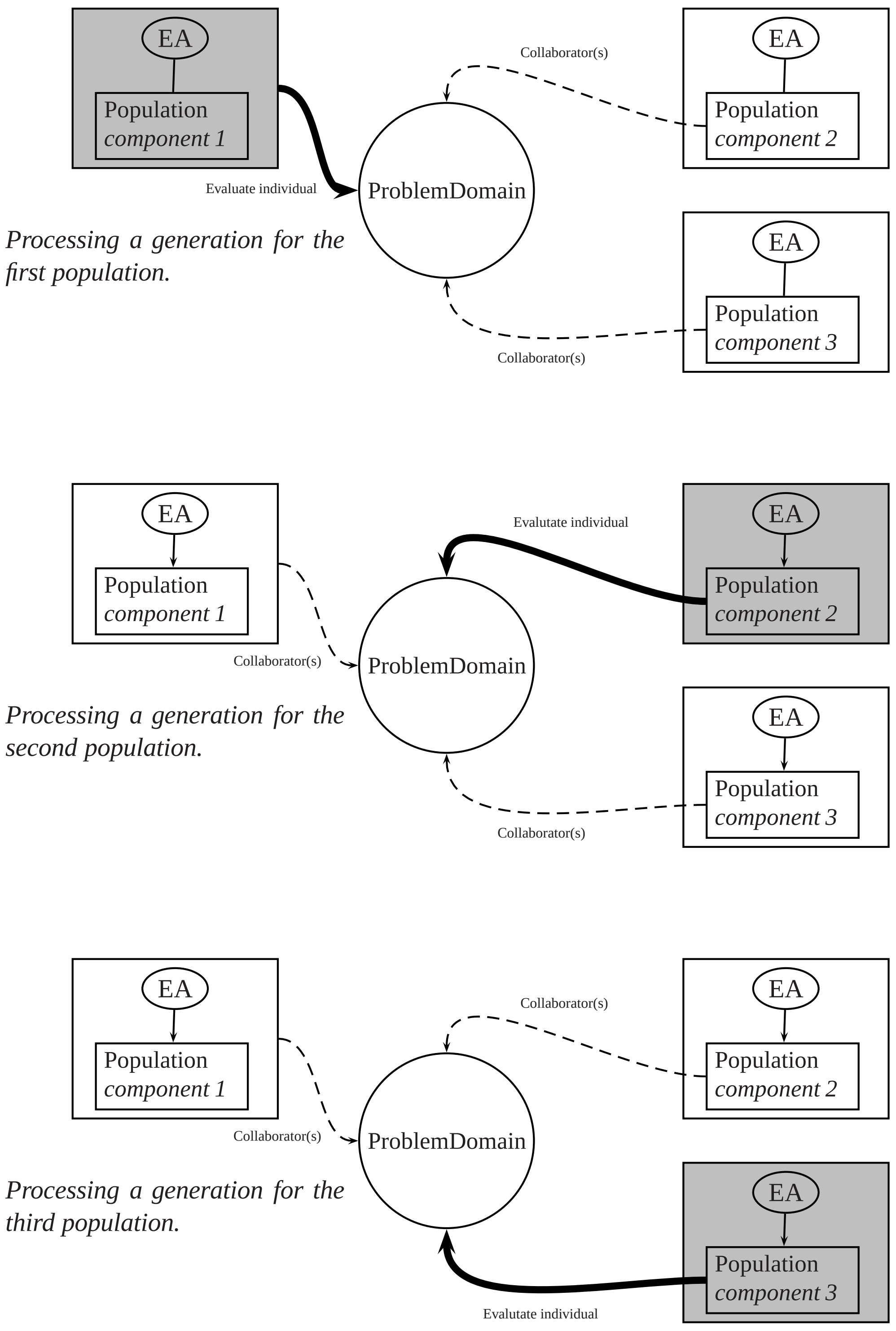


Figure 3.2: Illustration of a round of a three population CCEA.

3.3.1 Decomposing a Static Function Optimization Problem

The problem class defined above is attractive in part because of its simplicity. It is also attractive because despite its simplicity there remain interesting research issues surrounding how one decomposes the domain such to be represented by multiple populations. Moreover, since a large number of real world applications fall into this domain, answering questions about static optimization provides useful information for practitioners.

In essence, the decomposition issue is a question of how to partition D. Consider, for example, the domain consisting of a vector of arguments in Euclidean space ($\vec{x} \in \mathbb{R}^n$). Here each argument of the function might be represented by a different population. In such a case, evaluating an individual from the first first population (argument x_1, for example), requires choosing collaborators from the other populations representing other arguments (arguments x_2 and x_3, for example). Figure 3.3 on page 31 illustrates this case.

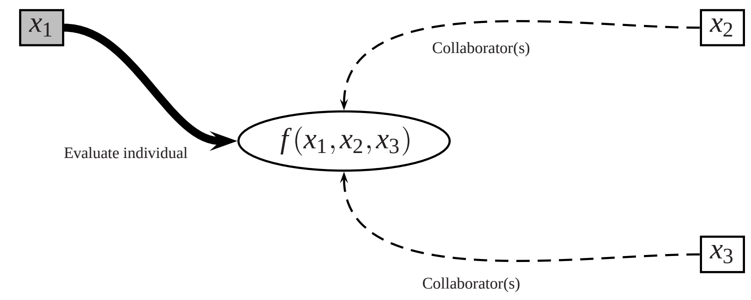


Figure 3.3: Illustration of how a CCEA might assemble a decomposed real-valued argument list to evaluate against a static objective function.

In the case of a binary representation, such a partitioning is most obviously done by dividing the binary string into different substrings. Individuals in each population might represent a particular partitioned substring, and evaluation of an individual substring will require representation from substrings in other populations. For example, Figure 3.4 offers an example to illustrate how a decomposed binary string might be assembled in order to evaluate it with some objective function. In this particular case, the string is broken up into three equal length segments for each population. This is only one choice. Part of the analysis in the next chapter will deal with exactly this choice.

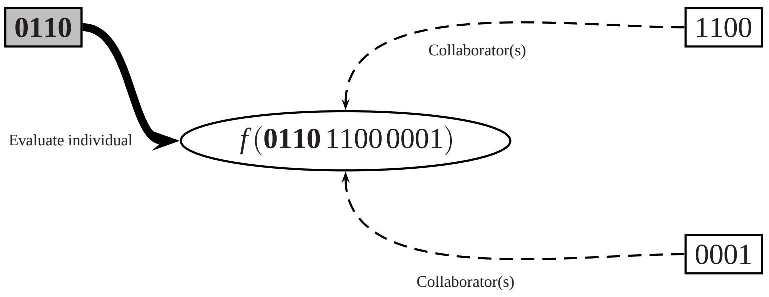


Figure 3.4: Illustration of how a CCEA might assemble a decomposed binary to evaluate against a static objective function.

3.3.2 Symmetry

The field of game theory (GT) has a useful notion of *symmetry* with respect to games. In this context, a symmetric game is one in which the reward gained for a particular set of strategies, as defined by each player picking a specific strategy, will be the same, regardless of which player is currently being considered. For example, if one player plays strategy *a* and the other player plays strategy *b*, the rewards to players depend only on *a* and *b*, but not on the identities of the players.

On a more mathematical level, let $\pi : S \times S \rightarrow \mathbb{R}$ define a function mapping strategies from each of two players to a payoff value. Let $\pi_1(a,b)$ be the payoff for player 1 when player 1 plays strategy *a* and player 2 plays *b*, and $\pi_2(a,b)$ be the payoff for player 2. When a game is symmetric, $\pi_1(b,a) = \pi_2(a,b)$.

Having the property of symmetry certainly simplifies a game in many ways. It is exactly this simplification that makes it a good property to consider when looking at things from first principles. It should be clear that the partitioning methods discussed in the previous subsection will yield problems with such a property. In essence, the class of CCEAs in which I am interested is symmetric when applied to single-objective, static optimization problems, as defined above.

This property has several benefits. First, it should now be clear that the multiple population CCEA defined above as applied to problems with this property of symmetry are cooperative, by virtually any reasonable definition. A population (player) cannot succeed unless his collaborators (partners) also succeed, and it fails when they fail. Second, because of this property of symmetry, a clear objective measure is possible. Since the goal is to optimize the function and the rewards are symmetric, there is no red queen problem because any progress against the external measure (the given problem) *is*, by definition, objective. This resolves several of the analytical challenges observed in so far and leaves me with a simple coevolutionary system and a straightforward objective: single-objective static maximization.

Chapter 4

CCEAs as Static Optimizers

Is this class of multi-population, symmetric cooperative coevolutionary algorithms useful as a class of function optimizers? In some philosophical sense, all algorithms are optimizers when they are applied to optimization problems; they may simply be quite poor optimizers, yielding low quality solutions or taking a far longer time to find the optimum than other, better suited algorithms might. Perhaps it is unhelpful to wax philosophic: if one applies the algorithms to optimization problems, they are optimizers, and our issue (as scientists and engineers) is how they perform and how well they function. Perhaps the "fundamental question" posed in Chapter 1, while well-intentioned, is misphrased.

In this chapter, I take this exact approach and rephrase the fundamental question to ask "How, and how well, do CCEAs act as function optimizers?" Of course, this is a very broad question, with questionable specific research potential, so it must be further refined to offer more specific opportunities to provide explicative results. I do this by concentrating on understanding what makes function optimization problems difficult for CCEAs, and what algorithmic design decisions might be used to counter these difficulties. By giving the fundamental question context in this way, one can apply analytical and empirical tools to get more specific, useful answers than might otherwise be possible.

The chapter begins by first discussing how CCEAs can be applied to the problem domain of static function optimization defined in the previous chapter. This section takes up a property commonly believed to cause difficulties for CCEAs (*cross-population epistasis*), defines this property, and describes the intuition and historical issues that have lead to the suppositions surrounding its effect on CCEAs. The section that follows describes what elements of CCEAs might plausibly lead to coevolutionary advantage over traditional EAs, then explores this advantage in the context of cross-population epistasis. This is done by conducting theoretical investigations of very simple CCEAs, as well as empirical analysis of the operators involved in somewhat more complicated CCEAs. The third section of the chapter examines what is perhaps the most principal component of CCEAs to deal with challenges posed by complicated problem domains: methods of interaction (collaboration). This is done via primarily empirical means. I use the final section to bridge to the next chapter by illustrating, both empirically and formally, some of the challenges that face CCEAs when they are applied to static optimization problems.

4.1 Applying CCEAs

4.1.1 General Problem Domain Framework

Before beginning any analysis of the algorithm of study, one should be careful to select a framework for the problem domain that facilitates analysis at some level. Such a domain should be simple enough to gain appropriate intuition, but flexible enough to allow for a thorough exploration of the relevant properties of interest. To this end, this section concentrates entirely on so–called *pseudo-boolean* functions, $f : \{0,1\}^n \to \mathbb{R}$. Individuals will therefore maintain binary representations throughout this chapter. The use of binary representation (in this case) provides a greater intuition about the effects of the genetic operators—something that will come in handy during analysis.

Pseudo-boolean functions provides several additional advantages. First and foremost, they will make it very easy to describe the salient problem property that constitutes the focus of the research discussed in this section. Second, they make it very straightforward to construct examples (or counter examples) illustrating certain very explicit ideas relating to properties of the problem. Finally, the issue of how well suited a static decomposition is for the overall problem representation becomes simplified by the ease with which clear definitions of such partitioning may be established.

Basic Example Functions

Since they will be useful throughout this chapter, several simple pseudo-Boolean functions are defined below: ONEMAX, LEADINGONES, and TRAP. These are very common EA problems (Mitchell 1997). ONEMAX simply counts the number of ones in a given binary string, LEADINGONES counts the number of consecutive ones at the start of a string, and TRAP essentially establishes a false ONEMAX gradient leading away from the true maximum, found at the all 0 string. These are defined formally below.

Definition 8. *The function* ONEMAX: $\{0,1\}^n \rightarrow \mathbb{R}$ *is defined by*

$$\text{ONEMAX}(x) := \sum_{i=1}^{n} x_i$$

.

Definition 9. *The function* LEADINGONES: $\{0,1\}^n \rightarrow \mathbb{R}$ *is defined by*

$$\text{LEADINGONES}(x) := \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$$

.

Definition 10. *The function* TRAP: $\{0,1\}^n \rightarrow \mathbb{R}$ *is defined by*

$$\text{TRAP}(x) := \left((n+1) \cdot \prod_{i=1}^{n} x_i \right) + n - \text{ONEMAX}(x)$$

.

Using these simple functions, somewhat more complicated functions can be constructed. In fact the ONEMAX problem can be seen as a specific case of so-called LINEAR functions. Additionally, it is possible to generalize the LEADINGONES problem to one in which only b blocks ones may be counted at the start of the string, LOB_b (Jansen and Wiegand 2003a). The following definitions for these two problem classes are offered below.

Definition 11. *The function* ONEMAX: $\{0,1\}^n \rightarrow \mathbb{R}$, *and some vector* $w \in \mathbb{R}^n$ *is defined by*

$$\text{LINEAR}(x) := \sum_{i=1}^{n} w_i x_i$$

.

Definition 12. *For* $n \in \mathbb{N}$ *and* $b \in \{1, \ldots, n\}$ *with* $n/b \in \mathbb{N}$, *the function* LOB_b: $\{0,1\}^n \rightarrow \mathbb{R}$ *(short for LeadingOnesBlocks) is defined by*

$$\text{LOB}_b(x) := \sum_{i=1}^{n/b} \prod_{j=1}^{b \cdot i} x_j$$

.

Separability

An important property of problem domains in general is the notion of *separability*. Separability refers to the ways in which one can partition the string so that the total objective function value is merely the sum of functions operating over disjoint substrings. A formal definition follows.

Definition 13. *A function* $f\colon \{0,1\}^n \to \mathbb{R}$ *is called* (r,s)-separable, *where* $r,s \in \{1,2,\ldots,n\}$, *if there exists a partition of* $\{1,\ldots,n\}$ *into r disjoint sets* $I_1,\ldots,I_r$ *with* $I_j = \left\{i_{j,1},\ldots,i_{j,\|I_j\|}\right\}$ *and* $\|I_j\| \leq s$*, for all* $j \in \{1,\ldots,r\}$*, and if there exist corresponding pseudo-Boolean functions* $g_1,\ldots,g_r$ *with* $g_j\colon \{0,1\}^{\|I_j\|} \to \mathbb{R}$ *such that the following holds:*

$$\forall x = x_1 \ldots x_n \in \{0,1\}^n\colon\ f(x) = \sum_{j=1}^{r} g_j\left(x_{i_{j,1}} x_{i_{j,2}} \cdots x_{i_{j,\|I_j\|}}\right)$$

A function that is (r,s)-separable is one that can be broken down into r linearly independent substrings of length no greater than s. The function f is *exactly* (r,s)*-separable* if f is (r,s)-separable but not (r',s')-separable for any $r' > r$ or $s' < s$. So, for example, the OneMax and Linear functions are exactly $(n,1)$-separable (or *fully separable*), whereas the LeadingOnes and Trap functions are exactly $(1,n)$-separable (or *inseparable*).

During the course of analysis, it will be necessary to construct problems with some specific degree of separability. There is a common and straightforward way to do this using a tactic called *concatenation*. A *concatenated* function is one in which the binary string is partitioned in some way, and the objective function value is the linear sum of the underlying subordinate function applied to the individual substrings. Typically, as well as in my case, the underlying function itself is inseparable, so the constructed function is then exactly separated by the number of concatenated pieces. More directly, one simply defines a function in terms of the sum of r independent sub-functions, which are themselves inseparable. A simple example is obtained by choosing an inseparable function of s bits and concatenating r of these s-bit blocks to form an rs-bit (where $n = rs$) problem the value of which is just the sum of the individual functional pieces.

4.1.2 Representing Optimization Problems

Given the choice for a binary representation and pseudo-Boolean functions, what remains to be clarified regarding representation is how problem components will be divided. To start with, let me begin by assuming an even division of bits per population. That is, given a potential solution to f, a bit string $x \in \{0,1\}^n$, and k populations, x is divided into k separate components of equal size $l = n/k$. How those bits are assigned will be the subject of study later in this section, but unless otherwise stated it should be assumed to be consecutive. That is, assuming $x = x^{(1)}x^{(2)}\ldots x^{(k)}$, where $\|x^{(i)}\| = l$ is the length of each piece, and $i \in \{1,\ldots,k\}$ represents an index indicating from which population the substring was drawn. Therefore, I can define the substring $x^{(i)} = x_{(i-1)\cdot l+1}\cdots x_{i\cdot l}$ from the total bit string. A diagram of this is shown below.

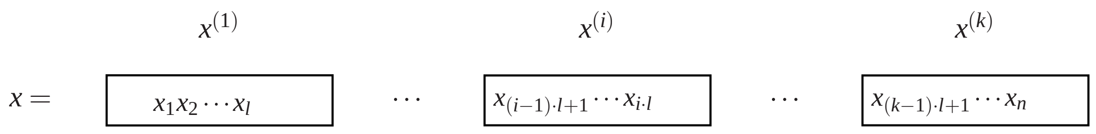

$x^{(1)}$
$x^{(i)}$
$x^{(k)}$
$x =$
$x_1 x_2 \cdots x_l$
$\cdots$
$x_{(i-1)\cdot l+1} \cdots x_{i\cdot l}$
$\cdots$
$x_{(k-1)\cdot l+1} \cdots x_n$

Figure 4.1: This figure illustrates how a binary string is partitioned into consecutive substrings, as well as showing how the notation used relates to this partitioning.

Here each population is assigned to one of the *k* substrings, and individuals from that population represent potential values for that role. During the collaboration process a total string is assembled and evaluated against the objective function. As discussed in the last chapter, this allows for a very straightforward encoding of the values for the problem into the representation, and decoding of the represented solutions in order to evaluate them in the context of the complete problem.

4.1.3 Cross-population Epistasis

What is omitted from the previous subsection is a much wider issue: *in general* how should an engineer statically decompose a problem *a priori*? A partitioning method like the one above may seem naive, perhaps even myopic. What are the properties of the problem? How can one tell whether the problem has been "appropriately divided" or not?

The presence of non-linear relationships between genes,or *epistasis*, has been an important part of EC research (Goldberg 1989). In cooperative coevolution the division of the problem into components generates a new complexity to consider: whether or not such non-linearities cross population boundaries. Indeed, various researchers working on cooperative coevolutionary algorithms have attempted to understand the effects of non-linear relationships between components (Wiegand, Liles, and De Jong 2002a; Bull 2001; Wiegand, Liles, and De Jong 2001; Bull 1997; Potter and De Jong 1994). The prevailing belief is that the existence of cross-population epistasis poses particular difficulties for CCEAs. There is a certain amount of satisfying intuition to justify this belief. If an EA's job is to assemble lower-order blocks into increasingly higher-order blocks then presumably splitting up a function in such a way that groups have little to no non-linear interactions between them would seem to give the EA an advantage. At the same time, divisions that preserve or create non-linear relationships across the population boundaries would seem to pose potential difficulties to CCEAs. The question of the existence or absence of cross-population epistasis is one of the main properties under investigation in this chapter. Before this can be done, a more thorough understanding of what this question is really asking and what the terms really mean must be undertaken.

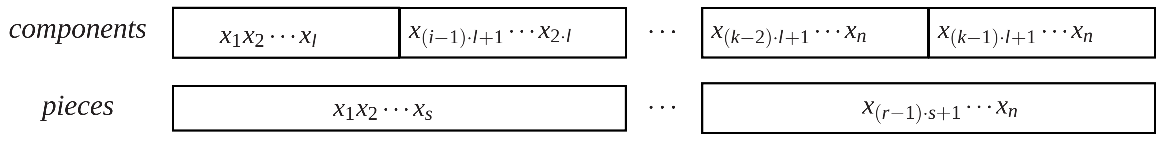
While it is clear that one can divide the problem any way one wishes, there may be more "natural" places to divide it than others. By "natural" I mean divisions that correspond to the notions of linearity versus non-linearity across population boundaries. This is, of course, essentially addressing the issue of the separability of the function. Indeed, as engineers we are concerned with the degree to which an algorithm's decomposition may be closely aligned with the problem's true separation. I use the term *match* to distinguish between such cases. When the problem pieces are separable between the components represented in the populations, I say that the decomposition *matches* the separability of the problem. To be more specific, I say that the decomposition *exactly matches* the separability of the problem when the pieces are exactly separable. This is defined more formally below.

Definition 14. *Let a function* $f : \{0,1\}^n \to \mathbb{R}$ *be* (r,s)*-separable as in Definition 13. A decomposition* matches *the separability of* f *if all bits that belong to one index set* I_j *are in one population.*

The decomposition exactly matches *the separability of* f *if there are* r *components, and each EA operates on the bits contained in exactly one of the index sets* I_j.

The phrasing used until this point reflects opposite sides of the issue of matching. An algorithm that has a decomposition that matches the problem can be said to be "separable across population boundaries", while one that does not match exhibits "cross-population epistasis" in the sense that there exist nonlinear relationships between components resulting from decompositions that place portions of inseparable pieces in different populations. It should be clear that when decompositions do not match the problem's separability, there will be such nonlinearities by definition.

An illustration will clarify the differences between components and pieces, as well as separability and epistasis. Starting with notation, I will use k to mean the number of populations, or EAs. This corresponds to the division of the problem created by the algorithm's designer into distinct components, typically all of the same length, l. The true problem is divided into r distinct, inseparable pieces, each of length s. Figure 4.2 below illustrates an example where these two do not match.



components
$x_1x_2\cdots x_l$
$x_{(i-1)\cdot l+1}\cdots x_{2\cdot l}$
$\cdots$
$x_{(k-2)\cdot l+1}\cdots x_n$
$x_{(k-1)\cdot l+1}\cdots x_n$
pieces
$x_1x_2\cdots x_s$
$\cdots$
$x_{(r-1)\cdot s+1}\cdots x_n$

Figure 4.2: This figure illustrates how differences between divisions chosen by the algorithm designer (components) and the natural separations of the problem (pieces) can be mismatched.

4.2 Partitioning and Focussing

Perhaps the best place to start to address the reformulated fundamental question is to investigate two specific questions: "can cooperative coevolution perform better than a comparable traditional EA on static optimization problems" and, if so, "what properties of CCEAs lead to this increased performance"? As the reader will see (and as is unsurprising), cooperative coevolution can perform better or worse than traditional evolution, depending on properties of the problem. In fact, it is the second question that holds more interest: what is the CCEA advantage?

One nice thing about analyzing this particular class of CEAs (cooperative symmetric multi-population EAs) is that they are not so fundamentally different from traditional EAs that one cannot see and grasp (at least at a high level) the differences between the two fairly simply. Indeed, there seem to be only two real augmentations to these CCEAs that offer potential for outperforming more traditional EAs (in certain circumstances): their ability to divide the problem into smaller subspaces in which to search and their ability to increase the effects of the operators on those smaller components. This can be seen as a kind of partitioning and focussing; that is, exploration of the components can be increased without as much risk of disruption of the entire candidate solution. Understanding the nature of these two properties, partitioning and focussing, is the point of this section.

4.2.1 Run Time Analysis

When one is interested in understanding how an algorithm performs, traditional computer science would suggest an analysis of run time behavior in the context of canonical complexity bounding functions (Cormen, Leiserson, Rivest, and Stein 2001). However, EAs are very complicated algorithms offering much that creates difficulty for run time analysis, CEAs are likely to be even more difficult. Nevertheless, a host tools available for analysis of randomized algorithms (Motwani and Raghavan 2000) provide methods for learning something about relatively simple EAs.

Analysis of this sort typically works by attempting to answer the question: "How many function evaluations does one expect to compute, on average, before the first time the global optimum is reached?" Typically the algorithm is considered to run forever, and the expected *first hitting time*, or *optimization time*, is computed—generally bounded by some probability measure. The results of such analysis are firm, trusted predictions of the order of the number of evaluations needed to find a solution as the problem scales in size.

This analysis provides a good place to start in terms of understanding wider classes of more complicated EAs. This subsection discusses some early analysis of a simple CCEA, indicating how and when the partitioning and focussing can give the CCEA an advantage. The majority of the formal analysis in this subsection was performed in collaboration with Thomas Jansen during a visit to Dortmund University. This visit was funded by the Deutsche Forschungsegemeinschaft as a part of the Collaborative Research Center "Computational Intelligence" (SFB 531), and much of the results of this work can be found in Jansen and Wiegand (2003b) and (Jansen and Wiegand 2003c).

The simple CCEA I will analyze here is formulated by using a known simple EA in the Potter framework described in the previous chapter.

The (1+1) EA

I begin by describing the simple, analytically tractable (1+1) EA, shown in Algorithm 3 below. Analysis of the (1+1) EA has provided the EC community with many results (Muehlenbein 1992; Rudolph 1997; Garnier, Kallel, and Schoenauer 1999; Droste, Jansen, and Wegener 2002; Scharnow, Tinnefeld, and Wegener 2002), as well as tools and methods that could prove useful for analyzing more complicated EAs (Wegener 2002). This simple EA serves the purposes of analysis well, since it is very simple but still quite similar to many more complicated EAs in many ways. What follows is a formal definition of the (1+1) EA as applied to the maximization of pseudo-Boolean functions. The reader should note that this is a specific case of Algorithm 1.

Algorithm 3 (1+1 Evolutionary Algorithm ((1+1) EA)).

```
1. Initialize population
   Choose $x_0 \in \{0,1\}^n$ uniformly at random.
2. Evaluate population
   $f_p = f(x_0)$
3. $t := 0$
4. do
      4.1 Select parents from population
          Use $x_t$ as parent
      4.2 Generate offspring from parents
          Create $y \in \{0,1\}^n$ by copying $x_t$ and, independently for each bit,
          flip this bit with probability $\min\{1/n, 1/2\}$.
      4.3 Evaluate offspring
          $f_c = f(y)$
      4.4 Select survivors for new population
          If $f_c \geq f_p$, then set $x_{t+1} := y$, else set $x_{t+1} := x_t$.
      4.5 t := t + 1
   until Terminating criteria is met.
```

This algorithm can be described very simply. There is only one individual in the parent population. The offspring is generated by bit-flip mutation on the parent at a rate of $1/n$, though the mutation probability cannot exceed 1/2. The offspring and parent compete for survival for the next generation, and the offspring survives if it is as least as good as the parent. It is, as I have said, a simple algorithm about which much is known in terms of run time performance. For example, for all linear functions with non-zero weights (such as ONEMAX), the running time is known to be $\Theta(n \ln n)$; the running time for the LEADINGONES problem is known to be $\Theta(n^2)$ (Droste, Jansen, and Wegener 2002).

The CC (1+1) EA

A simple CCEA can be produced by using the EA defined above in the Potter framework. Such an algorithm has many benefits for the purposes of analysis. For one thing, it is perhaps the simplest conceivable cooperative coevolutionary algorithm that can be applied to static optimization problems. This is an attractive property since, like the (1+1) EA, it shares some useful properties with more complicated CCEAs, while being analytically tractable. Another useful feature of this simplified algorithm is that the issue of how collaborators are selected is eschewed since the population size for each EA is only one, by definition. For pedantism, the CC (1+1) EA is described formally, in detail below in Algorithm 4. This is an augmentation of Algorithm 2 below.

Algorithm 4 (Sequential Cooperative Coevolutionary 1+1 EA (CC (1+1) EA)).

```
1. for $a := 1 \dots k$
   1.1 **Initialize** population a
       Choose $x_0^{(a)} \in \{0,1\}^l$ uniformly at random.
2. for $a := 1 \dots k$
   2.1 **Evaluate** population a
       Not applicable
3. $t := 0$
4. do
   4.1 for $a := 1 \dots k$
       4.1.1 **Select parents** from population a
             Use $x_t^{(a)}$ as parent
       4.1.2 **Generate offspring** from parents
             Create $y^{(a)} \in \{0,1\}^l$ by copying $x_t^{(a)}$ and, independently for each bit,
             flip this bit with probability $\min\{1/l, 1/2\}$.
       4.1.3 **Select collaborators** from population
             Use all current components of $x_t$, except $x_t^{(a)}$
       4.1.4 **Evaluate** offspring with collaborators
             $f_c^{(a)} = f\left(x_{t+1}^{(1)} \cdots y^{(a)} \cdots x_t^{(k)}\right)$
             $f_p^{(a)} = f\left(x_{t+1}^{(1)} \cdots x_t^{(a)} \cdots x_t^{(k)}\right)$
       4.1.5 **Select survivors** for new population a
             if $\left(f_c^{(a)} \geq f_p^{(a)}\right)$ set $x_{t+1}^{(a)} := y^{(a)}$
             else set $x_{t+1}^{(a)} := x_t^{(a)}$
       4.1.6 t := t + 1
until **Terminating criteria** is met.
```

There is only one parent component for each of the k populations. The algorithm works by sequentially moving from one population to the next, producing a mutated offspring component, and comparing it against the parent in the current context of all other components. If the offspring is at least as good as the parent, the offspring replaces the parent. It should be noted that this formulation of the CC (1+1) EA requires two evaluations for each generation (the parent must be evaluated for each new offspring component, since collaborating individuals may have changed from the previous round). Since this results in merely a constant factor in the analysis, this fact is ignored in this section.

Even at this simple level, there are a variety of choices that might have been made yielding slightly different algorithms. Predominantly these have to do with which generation provides information to the next. The differences between such "update mechanisms" is explored in more detail in the study presented by Jansen and Wiegand (2003c). In this work, the update mechanism is assumed to work as described above: each EA completes a generation with its respective population before the CCEA moves sequentially to the EA operating on the next component.

Comparing The (1+1) EA and The CC (1+1) EA (Separability)

Let us suppose for the sake of argument that the CCEA gains its advantage chiefly (if not exclusively) from the advantage of slicing the problem up and solving the pieces in parallel. This certainly seems to satiate the most straightforward intuition: if the subproblems are linearly independent, surely an algorithm that solves these independent subproblems independently in parallel has advantage over the algorithm that treats it as one big problem? After all, their relative search spaces would (presumably) be much smaller. As a vague sort of justification of this intuition, consider a two dimensional optimization problem $f(x,y) = g_1(x) + g_2(y)$, where $x \in \{a,b,c\}$ and $y \in \{d,e,f\}$. The algorithm that solves the full $f(x,y)$ must search a space with 9 search points, while splitting the problem up into two algorithms, each searching $g_1(x)$ and $g_2(y)$ respectively only need search three points each, or a total of 6 search points. As the size of the relative sets increases, this gap widens, of course.

But this is not the space that a CCEA searches, even under the most ideal circumstances, and this notion of subproblems that are separable across population boundaries (even if the design engineer “conveniently” is aware of such divisions in the problem) is also not the advantage it appears to be. To get a better understanding of this it is necessary to first contradict intuition by analyzing the most separable problems conceivable: LINEAR problems. If the mere splitting of the problem into appropriate partitions is sufficient to gain advantage, one would expect the biggest advantage to appear when the problem is most divisible. In fact, this is not the case. As Jansen and Wiegand (2003b) shows, the CC (1+1) EA is still bounded above by $O(n \ln n)$, yielding no advantage whatsoever. In fact, it is clear from the analysis produced in that work that no division of the problem will result in an improvement for the CC (1+1) EA over the simple (1+1) EA.

Why is this? Consider that a linear problem still requires no more than one-bit mutations for there to be improvements. Such being the case, the increased focus of attention by the mutation operator offered by the CCEA is not necessary, and cannot be leveraged over the traditional EA for such problems. In fact, for the CC (1+1) EA to show an advantage over the (1+1) EA, the problem must require *both* the partitioning *and* the increased focus of the operators. It must use the partitioning to *protect* the other pieces of the problem from the added disruption of the more attentive genetic operator. This is not the case in the linear problem.

A problem can be constructed to exhibit this property very clearly. Taking the LOB_b problem described above, and creating a problem that is separable across population boundaries by “concatenating” several of them together in a linear manner, it is easy to construct a problem that has exactly the property in which I am interested. Allowing, for the moment, the algorithm decomposition to exactly match this new problem’s true separability, I can define the new function $\mathrm{CLOB}_{b,k}$ in terms of the k populations in the CC (1+1) EA.

Definition 15. *For $n \in \mathbb{N}$, $k \in \{1,\dots,k\}$ with $n/k \in \mathbb{N}$, and $b \in \{1,\dots,n/k\}$ with $n/(bk) \in \mathbb{N}$, the function $\mathrm{CLOB}_{b,k} \colon \{0,1\}^n \to \mathbb{R}$ is defined by*

$$\mathrm{CLOB}_{b,k}(x) := \left(\sum_{h=1}^{k} n \cdot \mathrm{LOB}_b \left(x_{(h-1)l+1} \cdots x_{hl} \right) \right) - \mathrm{ONEMAX}(x)$$

for all $x = x_1 \cdots x_n \in \{0,1\}^n$, with $l := n/k$.

Analysis of both algorithms on this problem shows a clear separation between the CC (1+1) EA and the (1+1) EA, in terms of b and k. The expected optimization time of the CC (1+1) EA on the function $\mathrm{CLOB}_{b,k}$ is $\Theta\left(kl^b \left(\frac{l}{b} + \ln k\right)\right)$ with $l = n/k$, if the CC (1+1) EA exactly matches the function’s separability with k (1+1) EAs, and $2 \le b \le n/k$, $1 \le k \le n/4$, and $n/(bk) \in \mathbb{N}$ hold. The same problem requires the (1+1) EA $\Theta\left(n^b \left(n/(bk) + \ln k\right)\right)$ evaluations. Dividing the former by the latter, the difference is obtained and shown below.

$$\frac{\Theta\left(n^b \cdot \left(\frac{n}{bk} + \ln k\right)\right)}{\Theta\left(kl^b\left(\frac{l}{b} + \ln k\right)\right)} \quad = \quad \Theta\left(k^{b-1}\right)$$

It is interesting to note that for some values of k and b this can decrease the expected optimization time from super-polynomial for the (1+1) EA to polynomial for the CC (1+1) EA. This is, for example, the case for $k = n(\log\log n)/(2\log n)$ and $b = (\log n)/\log\log n$. The proofs for these assertions are demonstrated in detail in Jansen and Wiegand (2003b).

Comparing The (1+1) EA and The CC (1+1) EA (Inseparability)

Perhaps the reader is willing to accept that separation may be *insufficient* to guarantee that a CCEA will outperform an EA, but that *inseparability* constitutes a particular stumbling block for the algorithm. In fact this, too, need not be the case. This is clear when one considers the LEADINGONES problem. Simple analysis can show that this problem is still solved in quadratic time by both coevolutionary and non-coevolutionary variants of the (1+1) algorithms described above. Though the proof of this (shown below) is a bit tedious, it is nonetheless constructive and I will include it here for completeness.

Theorem 1. *The expected optimization time for the CC (1+1) EA on the function* LEADINGONES *is* $\Theta(n^2)$.

Proof. The proof for the upper bound is very straightforward. Let us pessimistically assume that the algorithm solves its components from left to right, one at a time. Droste, Jansen, and Wegener (2002) tells us that the expected number of active steps needed for a given component of length l to reach the all-onrd string can be bounded above by $2el^2$. Since a component is active every k steps, it will require a given component at most $2el^2k$ steps to reach 1^l. Components are solved one at a time, so the sum of the expectations can be used to find the expected waiting time for the entire process, $\sum_{i=1}^{k} 2el^2k = O(n^2)$. For the lower bound, begin by defining the term *progressive component* to mean the left-most component that is not the all one string. Note that all of the components to the right of the progressive component evolve without influencing the function value. Since the original string is drawn at random, and the mutation events are generated independently and uniformly at random, the result for each of these components to the right of the progressive component must contain random strings. Thus, the process can be treated as successive solutions to individual leading ones problems for each component. The lower bound for a given component can be obtained from Droste, Jansen, and Wegener (2002), except that now one must show that no advantage can be obtained from the partition.

First, a proof the lower bound for the case $l > 3$ must be provided. I define the term *head start bits* to be the number of bits that lead a component when that component first becomes the progressive component. Since the string is random until the component becomes progressive, the expected number of such bits can be bound above by $l/3$. Chernoff bounds indicates that the probability that there are more than $\frac{2}{3}l$ head start bits is bounded above by $\frac{3}{4}^{l/3}$. Thus, from Droste, Jansen, and Wegener (2002) and this, the lower bound of the expected number of active steps is $\Omega(l - \frac{2}{3}l)^2) = \Omega(l^2/9)$. A step is active every k generations, there are $\Omega(kl^2/9)$ such generations. The algorithm can be slowed down by assuming that after reaching 1^l in the progressive component, there are no more mutations in the current round. Since each component becomes all one only once, this slows down the optimization by less than k^2. Now, in each round at most one component can be solved, which yields $\sum_{i=1}^{k} kl^2/9 = n^2/9$ as lower bound. Altogether this yields $n^2/9 - k^2 = \Omega(n^2)$ for $l > 3$.

For $l \leq 3$ it suffices to see that the expected number of leading ones gained in one round is constant. This implies that there are on average $\Omega(k) = \Omega(n)$ rounds before the optimum is reached. In each round there are $\Omega(n)$ function evaluations. □

Of course, the LEADINGONES problem is quite simple; however, it *is* fully inseparable, meaning it cannot be partitioned into any linearly independent pieces. Yet in spite of this, as can be seen above, there is no advantage to the traditional EA over the CCEA: in both cases the expected waiting time for the global optimum is quadratic. The algorithm's performance seems to be unaffected by the inseparability of the problem. The proof helps give us insight into at least two of the reasons for this. First of all, this problem must be solved in a specific order in terms of the bit positions in the string, regardless of which algorithm it uses. As a result, the partitioning makes very little difference to the problem's solution, one way or the other. Second, there is nothing intrinsically contradictory in the interrelated bits. The LEADINGONES function cannot drive the algorithm to a point where its partitioning makes it impossible to make the necessary jump to the solution. This notion of a certain *quality* or *characteristic property* of the cross-population epistatic linkages will become more important a little later in this chapter.

Not only is it the case that inseparability can pose *no* problem for the CCEA over the EA, it is not difficult to construct problems in which the advantage of the CCEA is preserved, despite the presence of cross-population epistasis. Moreover, such a problem can exhibit an exponential improvement of the CC (1+1) EA over the more traditional (1+1) EA. The reader is referred to Jansen and Wiegand (2003a) for more details; however, this result should not be surprising, given the *No Free Lunch Theorem* (Schumacher, Vose, and Whitley 2001; Wolpert and Macready 1997).

4.2.2 Empirical Analysis

Though asymptotic analysis provides exact answers to questions about the expected waiting time until the global optimum is reached, it does have its limitations. For one, there is a difference between an asymptotic result and the realistic conditions under which it can be observed. For another, the algorithms analyzed here were intentionally very simple, and it would be nice to know how the results carry over to somewhat more complicated algorithms that are more likely to be used by practitioners.

In fact, in this case the results discussed above hold for both realistic conditions of the (1+1) algorithms, as well as for somewhat more complicated and realistic algorithms. I will address both of these issue empirically below by considering their behaviors on the $\text{CLOB}_{b,k}$ and LEADINGONES problems.

Realistic Problem Conditions

With respect to the first point, knowing that there is an asymptotic difference between the (1+1) EA and the CC (1+1) EA on the $\text{CLOB}_{b,k}$ problem (as an example) doesn't tell us *when* that difference may become visible. If the constant factors are such that the coevolutionary advantage cannot be seen until $n \geq 1,000,000$, then the analysis may mean very little to people who are applying these methods to real problems. This is not the case though, as we will shortly see. The CC (1+1) EA advantage over the (1+1) EA is visible with very reasonable sizes of problem dimensionality.

I performed the following experiments, first on the $\text{CLOB}_{2,4}$ problem, then on the LEADINGONES problem. A single (1+1) EA and a CC (1+1) EA with $k = 4$ populations were compared. The algorithms were run in each case until the global optimum was found, and the number of function evaluations necessary to achieve this was tracked and reported.

In the case of the $\text{CLOB}_{2,4}$ problem, the algorithm's representational decomposition matched the problem's decomposition. Thus, there were $k = 4$ components and $s = 4$ pieces; however, the size of the components was varied, $l = \{2,4,6,8,10,12,14,16\}$ $(n = \{8,16,24,32,40,48,56,64\})$. In the LEADINGONES problem the same
$(n = \{8,16,24,32,40,48,56,64\})$ values were used. This yielded at total of 32 experimental groups (8 for each problem, for each group). All groups were run for 50 independent trials.

The results of the $\text{CLOB}_{2,4}$ experiments are not surprising. Figure 4.2.2 on page 44 shows that the CC (1+1) EA performs increasingly fewer evaluations to reach the global peak than its EA counterpart. This difference is statistically significant for $l \geq 2$ at a 95% confidence using t-tests and the Bonferoni adjustment (Hancock and Klockars 1996). On the other hand, Figure 4.2.2 (same page) shows no significant difference (either visually or statistically) between the two algorithms over the same range of n values. This indirectly validates the theoretical results directly in realistic sizes dimensions of the search space: the coevolutionary variant is superior to the traditional variant on the separable $\text{CLOB}_{2,4}$ problem, but they are indistinguishable (in terms of performance) on the inseparable LEADINGONES problem for all $n > 8$. In both cases a more direct validation would be achieved by considering the ratio of the curves and determining whether this ratio was non-constant. To do this, experiments with larger values of n would need to be run.

Realistic Algorithms

One complaint that has been leveled against the analytical work that has focussed on the (1+1) EA (and, by extension, this work focussing on the CC (1+1) EA), is the fact that it does not appear to be a “realistic” algorithm in the sense that it is not one that practitioners are likely to use. While a comprehensive study of a wide catalog of coevolutionary algorithms is outside the scope and point of this dissertation, there are some natural steps that can be made to make this algorithm a bit more realistic. Here I describe one such step and conduct the same empirical studies as were presented above on these more plausibly practical EAs.

The non-coevolutionary EA works as follows. There is a population of parents (in this case 10), and at each step a single offspring is produced by fitness proportionate selection and bit-flip mutation, with a probability of $1/n$, where n is the length of the bit string. The offspring is then evaluated and compared with parent population, replacing the worst member if it is at least as good. If there are more than one such parents, then one is selected at random for deletion. This is the so-called *steady state* EA (De Jong 1975). The reader should be careful to note that this algorithm is not too terribly different than the (1+1) EA. In a sense, one could consider it a modified (μ+1) EA, where proportionate parent selection is used rather than uniform random selection. This is an intentional choice, as it lends itself well toward future expansions of the theoretical analysis. I will refer to this as the SS-EA

The coevolutionary variant of this should be of no surprise. In this case, there are k (again here $k = 4$) populations, each of size 10. As with the sequential CC (1+1) EA, the populations of this SS-CCEA are processed sequentially, in order. In each case, one step of the underlying steady state EA is performed. Now that there is a population, though, I must address the issue of collaboration. For now, I will simply pick a single individual from each of the other populations that has the highest fitness from the last time that population was processed (the so-called *single-best* collaboration strategy).

Again I consider the $\text{CLOB}_{2,4}$ and LEADINGONES problems with the same experimental groups: $l = \{2,4,6,8,10,12,14,16\}$ for $\text{CLOB}_{2,4}$ and $n = \{8,16,24,32,40,48,56,64\}$ for LEADINGONES. Again all 32 groups were for 50 independent trials.

The results of the $\text{CLOB}_{2,4}$ experiments are consistent with those of the simpler (1+1) variants. Figure 4.2.2 on page 45 shows that the SS-CCEA still statistically significantly outperforms its EA counterpart (95% confidence using t-tests and the Bonferoni adjustment), though for the $l \geq 2$ groups now. Additionally, Figure 4.2.2 (same page) still shows no significant difference between the two algorithms over the same range of n values. This justifies, to some extent, the use of the simple algorithm: the theoretical results for the simpler (1+1) algorithms seem to hold for similar population-based approaches. Here again the values of n are too small to see non-constant ratios between the the curves.

4.2.3 The Parallel CC (1+1) EA

The above analysis reflects consideration of Algorithm 4 described on page 39. As I’ve already mentioned, this algorithm is *sequential* in the sense the individual components are processed one at a time, in order. This allows for changes in an individual occurring during a round to affect the fitness of individuals representing different components evaluated later that same round. Alternatively, one might apply a more *parallel* updating

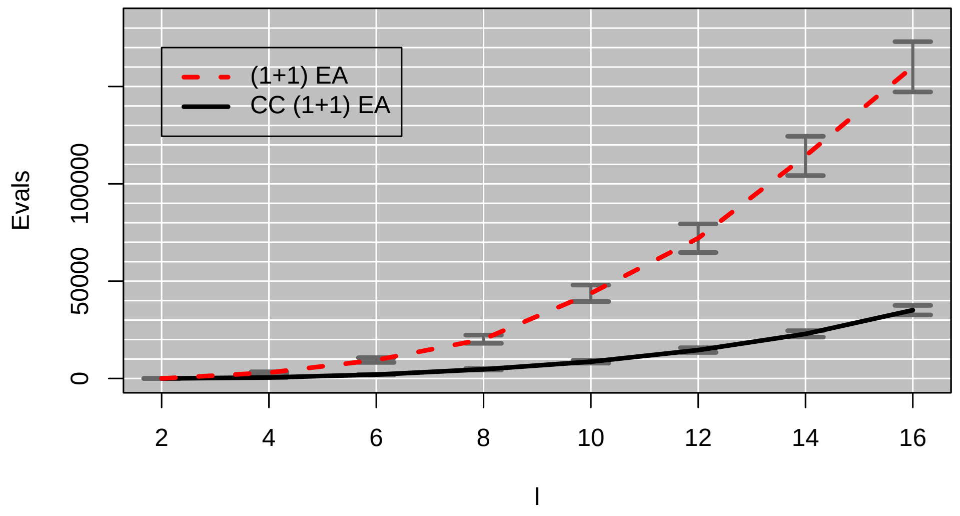


Figure 4.3: Results for the CC (1+1) EA (black) and the (1+1) EA (dashed, dark grey) on the CLOB$_{2,4}$ problem. The *x*-axis shows the length of a component, *l*, and the the *y*-axis shows the number of function evaluations until the global optimum was reached. The vertical wings show the 95% confidence interval of 50 independent trials; the plotted point shows the mean.

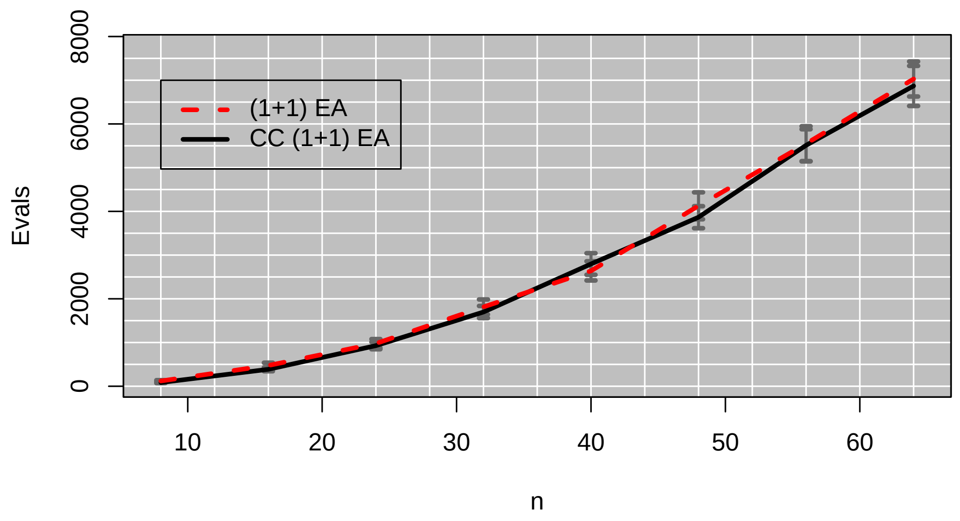


Figure 4.4: Results for the CC (1+1) EA (black) and the (1+1) EA (dashed, dark grey) on the LEADINGONES problem. The *x*-axis shows the length of a component, *l*, and the the *y*-axis shows the number of function evaluations until the global optimum was reached. The vertical wings show the 95% confidence interval of 50 independent trials; the plotted point shows the mean.

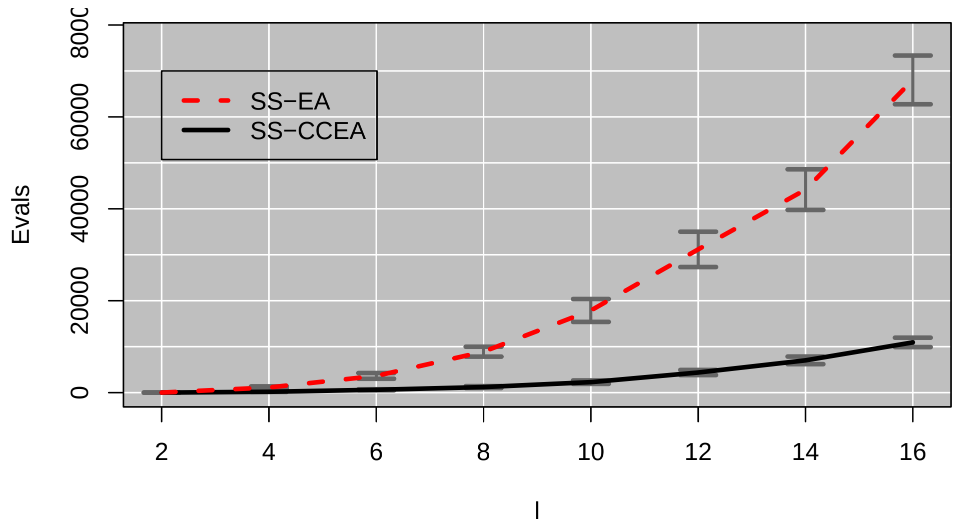


Figure 4.5: Results for the SS-CCEA (black) and the SS-EA (dashed, dark grey) on the CLOB$_{2,4}$ problem. The x-axis shows the length of a component /piece, l, and the the y-axis shows the number of function evaluations until the global optimum was reached. The vertical wings show the 95% confidence interval of 50 independent trials; the plotted point shows the mean.

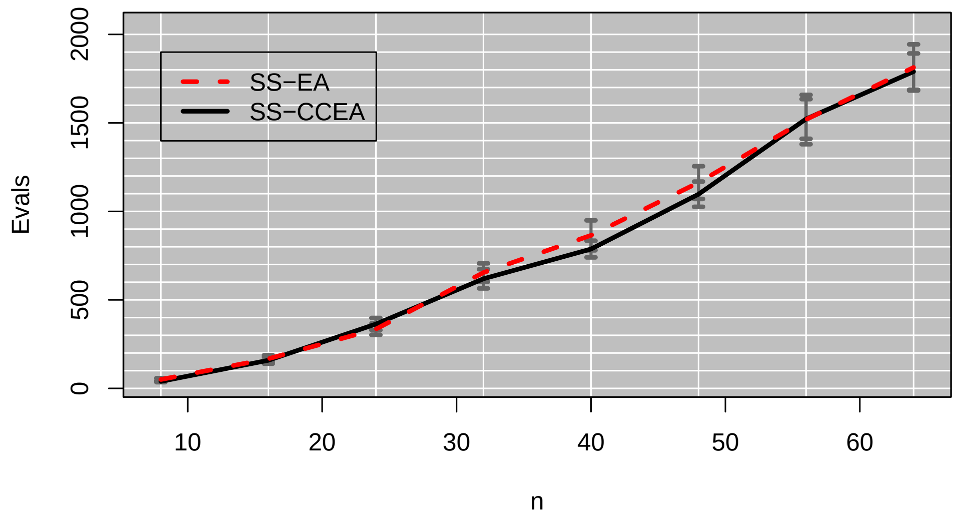


Figure 4.6: Results for the SS-CCEA (black) and the SS-EA (dashed, dark grey) on the LEADINGONES problem. The x-axis shows the length of a component /piece, l, and the the y-axis shows the number of function evaluations until the global optimum was reached. The vertical wings show the 95% confidence interval of 50 independent trials; the plotted point shows the mean.

approach, where individuals select collaborators from the previous round, allowing no inter-round changes to have any affect on the fitness result from individuals evaluated in the current round. I call this parallel because the components can then be processed asynchronously in any order, or they can be processed in parallel. Algorithm 5 can be obtained by simply making a slight change in line 4.1.4 of Algorithm 4. This substituted line is shown below.

Algorithm 5 (Parallel CC (1+1) EA).

```
    4.1.4  Evaluate offspring with collaborators
           $f_c^{(a)} = f\left(x_t^{(1)} \cdots y^{(a)} \cdots x_t^{(k)}\right)$
           $f_p^{(a)} = f\left(x_t^{(1)} \cdots x_t^{(a)} \cdots x_t^{(k)}\right)$
```

The reader should note that using the previous round's individuals for the evaluation affects not only the offspring evaluation, but the parent's as well. In fact, in the parallel variant one need only evaluate the parent once a round, since its value will never change while the round is in progress. This gives the parallel implementation an implicit advantage in terms of the number of evaluations performed, albeit by at most the constant factor 2.

The situations in which these two variants differ are perhaps less than obvious. Let's first look at the situation where problems are separable across population boundaries, then turn attention to a problem constructed to exploit the difference between these two algorithms. I will return to this algorithm at the end of the chapter to discuss some of its pathologies compared to those of the sequential model.

Equivalence on Separable Functions

Conventional wisdom suggests that, while these two algorithms are very similar, they make a fundamentally different type of choice with respect to selection and updating, and thus can differ profoundly from one another. While this is true in the general case, it is not true for problems in which components relate in merely linear ways. Indeed, the parallel and sequential variants of the CC (1+1) EA have equivalent running times on all functions that are separable across population boundaries. The proof of this is shown below for Theorem 2.

The result of this is to suggest a far more subtle difference between these two algorithms. Certainly they differ, and thus must exhibit different running times for some problems, but not for separable ones. This analysis suggests that if an algorithm designer is confident that their decomposition matches that of the true problem, there is no reason based on running time alone to choose the sequential algorithm over the parallel, or vice-versa. Thus they are free to consider other factors in making this decision (such as whether or not a parallel computational environment exists of which they can take advantage, for instance).

Theorem 2. *For any function $f\colon \{0,1\}^n \to \mathbb{R}$, if the parallel and sequential forms of the CC (1+1) EA use the same division into components on f and if f is separable across population boundaries, then the two CC (1+1) EAs behave identically.*

Proof. Contradiction is used to show that the two algorithms must have the same performance because they behave identically in all cases when applied to the same separable problem. Given that they behave identically, the number of generations needed for optimization is equivalent, and their respective expected waiting times for the global optimum cannot differ by more than a factor of two.

For there to be any difference in behaviors, the algorithms must make different decisions during the selection step, and by consequence the evaluation step, step 4.1.4, must obtain different results. Given that the function is separable across population boundaries, the linear contribution of the active component to the total fitness, $g_a : \{0,1\}^l \to \mathbb{R}$, can be the only difference in the total fitness score. As a result, for a different selection decision to be made, it must be true that $g_a(y^{(a)}) \geq g_a(x_t^{(a)})$ and $g_a(y^{(a)}) < g_a(x_t^{(a)})$. Since both inequalities cannot be simultaneously true, there is a contradiction. $\square$

Tricking The Sequential (or Parallel) Algorithm

Despite the fact that the parallel and sequential CC (1+1) EAs are very similar algorithms, and despite the fact that they perform identically on problems that are separable across population boundaries, they are nevertheless different algorithms and behave differently on some problems. The problem classes for which a distinction in running time is possible are a subset of problems containing cross-population epistatic linkages; however, there is no reason to believe that all such problems pose an advantage to one or the other—quite the reverse: some problems favor sequential algorithms and some favor parallel algorithms. Let's begin by looking at the latter.

When might a parallel algorithm gain advantage over the sequential algorithm? To see this, first note that the parallel algorithm actually *throws information away* in the sense that information might be learned during a round, but cannot be used until the next round, after other changes may have occurred. It makes sense, then, that parallel algorithms are favored when such information is misleading in some way. With a bit of effort, such a situation can be constructed. Consider the definitions below.

Definition 16. *For any $m \in \mathbb{N}$ with $m \geq 6$, we define $n := 2m$ and the function $g\colon \{0,1\}^n \to \mathbb{R}$. For $x = x_1 \cdots x_n \in \{0,1\}^n$ the x string can be written $x = x'x''$ with $x' = x_1 \cdots x_m \in \{0,1\}^m$ and $x'' = x_{m+1} \cdots x_n \in \{0,1\}^m$. The function g is defined by*

$$g(x) := \begin{cases} 2n+2 & \text{if } x = 0^n \\ 2n + n \cdot \|x\| & \text{if } x' = 0^m \ \vee \ x' = 0^m \\ 2n+1 & \text{if } x' \neq 0^m \ \wedge \ x'' \neq 0^m \ \wedge \ \|x\| = 2 \\ 2n + 2 \cdot \|x\| & \begin{array}{l}\text{if } x' \neq 0^m \ \wedge \ x'' \neq 0^m \ \wedge \ \|x\| > 2 \ \wedge \\ (\|x'\| < \lceil m/3 \rceil \ \vee \ \|x''\| < \lceil m/3 \rceil)\end{array} \\ n^4 + \|x\| & \text{otherwise} \end{cases}$$

for all $x \in \{0,1\}^n$, where $\|x\| = \textsc{OneMax}(x)$.

This function, $g\colon \{0,1\}^n \to \mathbb{R}$, is not (r,s)-separable for any $s < n$. What is of interest is the performance of the two CC (1+1) EA variants on g when using two sub-populations, one operating on x' and the other operating on x''. These two parts are non-linearly related, but splitting them up in this way will facilitate opportunities for the parallel and sequential algorithms to come to different conclusions based on the information they have at their disposal when processing a given generation. One more layer is needed to complete this picture: let's embed g into another function, $\textsc{SepFun}$, the target of our analysis. Now the difference between these two variants is clear and provable.

Definition 17. *For any $k \in \mathbb{N}$ with $k \geq 6$, define $n := 4k^2$, $m := n/2 = 2k^2$ and the function $\textsc{SepFun}\colon \{0,1\}^n \to \mathbb{R}$. For $x = x_1 \cdots x_n \in \{0,1\}$, the x string can be written $x = x'x''$ with $x' = x_1 \cdots x_m \in \{0,1\}^m$ and $x'' = x_{m+1} \cdots x_n \in \{0,1\}^m$. For $x' = x_1 \cdots x_m \in \{0,1\}^m$, the x' substring can be written $x' = x^{(1)} x^{(2)} \cdots x^{(k)}$ with $x^{(i)} = x_{2(i-1)k+1} \cdots x_{2ik}$ for all $i \in \{1, \ldots, k\}$. The function $\textsc{SepFun}$ is defined by*

$$\textsc{SepFun}(x) := \begin{cases} n - \|x'\| + \textsc{LeadingOnes}(x'') & \text{if } x'' \neq 1^m \\ n + \sum_{i=1}^{k} g\left(x^{(i)}\right) & \text{otherwise} \end{cases}$$

for all $x \in \{0,1\}^n$, where $\|x\| = \textsc{OneMax}(x)$.

As I will discuss at the end of this chapter, neither the sequential nor the parallel algorithm are able to find the global max for some types of problems with cross-population epistasis. This is such a problem: there is no expectation that the CC (1+1) EA finds the global optimum of $\textsc{SepFun}$. Instead one must change perspective to one of approximation. I wish to know which of the two algorithms finds the larger function value in a specific number of generations, fixed in advance. Concentrating on a large but still polynomial numbers of generations, Jansen and Wiegand (2003c) prove that for the sequential algorithm, the best fitness is so-far discovered in t steps (where $t \geq 6n^{5/2} + 6n^2$ and $t = n^{O(1)}$) is $\frac{n^2}{4} + n^{3/2} + n$ with overwhelming probability. In the same number of steps, the parallel variant will have attained the n^4 case from g. Jansen and Wiegand (2003c) verify this experimentally, as well.

Perhaps more interesting than this, however, is the fact that a relatively obvious change in the underlying function reverses the situation. Consider an alternative g' below, substituted in SEPFUN for g.

Definition 18. *For any $m \in \mathbb{N}$ with $m \geq 6$, we define $n := 2m$ and the function $g' \colon \{0,1\}^n \to \mathbb{R}$.*

For $x = x_1 \cdots x_n \in \{0,1\}^n$, the x string can be written $x = x'x''$ with $x' = x_1 \cdots x_m \in \{0,1\}^m$ and $x'' = x_{m+1} \cdots x_n \in \{0,1\}^m$. The function g' is defined by

$$g'(x) := \begin{cases} 2n+2 & \text{if } x = 0^n \\ 2n + n\|x\| & \text{if } x' = 0^m \vee x'' = 0^m \\ 2n+1 & \text{if } x' \neq 0^m \wedge x'' \neq 0^m \wedge \|x\| = 2 \\ 2n + 2\|x\| & \text{if } x' \neq 0^m \wedge x'' \neq 0^m \wedge \|x\| > 2 \\ n^4 + \|x\| & \text{otherwise} \end{cases}$$

for all $x \in \{0,1\}^n$, where $\|x\| = \text{ONEMAX}(x)$.

Now the sequential implementation is clearly superior, because having
$x^{(i)} \in \left\{1^k0^k, 0^k1^k\right\}$ yields much better pay-off than $x^{(i)} = 1^{2k}$. Thus, while in some cases the parallel variant clearly performs better than the sequential algorithm, in other cases the reverse is true. The basic advantage of the parallel over the sequential CC (1+1) EA seems to center around whether accrual of local information during a round will be misleading, while this advantage is reversed if the omission of such information is misleading. Again there is no general reason (without domain knowledge) to apply one or the other type of algorithm.

4.3 Collaboration

Selecting a collaboration method for a CCEA is one of the most important decisions an engineer must make. Not only can such methods make or break the performance of real CCEAs on real problems, the available choices seem to be integrally tied to certain problem properties. Many researchers believe that the existence of cross-population epistasis will demand more sophisticated collaboration methods. This section presents empirical analysis of the collaboration mechanism from the perspective of this property.

4.3.1 Sampling Interactions

Independent EAs in a coevolutionary framework operates on components with incomplete information. Recall the thought exercise earlier where the search space seems to be reduced by splitting the problem into two parallel and independent searches of the linearly separable pieces of the problem. This idea is flawed because the coevolutionary algorithm is *not* only searching the projection of the problem. It is searching a projection of the problem *at one time*, and this projection *changes* throughout time, as the other populations change.

Collaboration is the process by which the population currently under evaluation collects information and the whole problem, about the relationship between components. Despite the fact that CCEAs will look only at a projection of the problem at any one time, the quality of that projection in terms of their ability to characterize the relevant features of the complete search space depends largely on the tractability of that characterization, as well as the ability of the collaboration process to gather useful information about the other populations.

The best way to think about the collaboration process is to consider it as a method for *sampling interaction space*. Given this view, how one selects collaborators is a major issue since the degree of complexity of the interaction space is complicated. It may be that more collaborators are needed per evaluation in order to better characterize the interaction space (a larger sample size), or that the *way* they are picked might lead to more efficient use of the samples one already has (a biased sample), etc. If one selects more collaborators than is necessary, search performance is harmed by unnecessary evaluations. In the worst case, complete mixing is used. In complete mixing all individuals are coupled with all individuals from all populations. This becomes combinatorially large as the number of populations increase linearly. However, if only a single individual from each population is sampled for collaboration, the algorithm may have insufficient information about the interaction space with which to construct a suitable search gradient. This problem is worse when such an individual is chosen poorly. The questions is: What kinds of sampling methods are most appropriate for problems with different properties?

4.3.2 Decompositional Bias

As we have already seen, different problems have different degrees of separability that may or may not correspond with particular CCEA representation choices. Since separability information is not generally available for difficult optimization problems, the focus here is on the case where there is a particular kind of mismatch between the CCEA representation and a problem's "natural" decomposition. The issue here is not whether such mismatches make the problem harder to solve. This will clearly happen for some problems (Potter and De Jong 1994; Bull 1997; Bull 2001; Wiegand, Liles, and De Jong 2001). Instead, the question is whether adopting more complex collaboration methods can alleviate such mismatches.

In order to answer this, it is useful to have problems that can be explicitly controlled in terms of their inherent separability. For pseudo-boolean functions this is not difficult using the method of concatenation already mentioned. From a practitioner's point of view, barring any problem specific domain knowledge, the simplest way to represent pseudo-boolean functions is to again break up the total bit string of length n into k equal sized components of length l and assign each component to a different population. Recall that r denotes the number of separable pieces of length s. Given these two different divisions, a decompositional mismatch of the representation may be due to *over-decomposition* ($k > r$) or *under-decomposition* ($k < r$). If there are more populations than there are separate pieces of the problem, there is likely to be strong interaction between populations in the system with respect to the problem, i.e., cross-population epistasis. If $k < r$, the advantage of the parallelism of coevolutionary search is sacrificed.

Again, our interest is whether difficulties due to the existence of decompositional bias can be alleviated by more complex collaboration methods. The motivation for this interest stems from an important observation: if there is no cross-population epistasis, a simple selection method for collaboration is sufficient. If two populations represent independent pieces of the problem, then optimizing one population independently of the other will result in the complete optimization of the problem. As long as the collaborator chosen for each population member is the same, it doesn't matter how one chooses it. However, it does matter how many collaborators one chooses, since picking more than one will incur more unnecessary evaluations. Therefore, in the absence of cross-population epistasis, selecting the single best individual (i.e., the most fit individual(s) in other population(s) from previous evaluation) from the other populations for collaboration is sufficient. In fact, one could pick this individual randomly, as long as it was the same individual for each member of the population during a given generation. The point isn't that any partnering scheme will result in a better collaboration than another, but that since each population can essentially be optimized independently, one only needs a *consistent* sample from which to establish collaboration.

So when would a more complicated collaboration selection method be needed? Recall that how one chooses collaborators essentially determines how one *samples the potential interactions* with the other population. There has to be a reason to believe that more than one sample is needed, or that sampling with a particular bias (say choosing the best) will result in a qualitatively different characterization of the relationship between the populations. Either way, some interaction between the populations is certainly needed to justify this need. More than simply having such epistasis is at issue, however.

In order to study the effects of collaboration on such situations, the following experiment was constructed.

The steady state CCEA (SS-CCEA) described above was applied to a concatenated LEADINGONES problem. In this particular case there were 128 bits in the total bit string of the problem, subdivided evenly into $r = 2$ pieces. The function below gives a more specific description of this function.

$$f(x) = \text{LEADINGONES}\left(x^{(1)}\right) + \text{LEADINGONES}\left(x^{(2)}\right)$$

A total of 6 collaboration selection methods were used. The number of collaborators chosen for a given evaluation was varied $(1,2,\&3)$ and two selection biases were used: c-best and c-random. These worked as follows. In the first case, the c best individuals were chosen from each of the other populations, evaluated with the current individual, and the best of those scores was used as the fitness value for the individual. In the second case, c individuals were chosen uniformly at random without replacement, evaluated with the current individual, and again the best of those scores were used as the fitness value for the individual.

Aside from these 6 collaboration methods, the number of populations, k, was varied. However, in all cases the number of individuals in each population was 10. The results for $k = 2$ through $k = 16$ in Table 4.1 show the average number of evaluations for the algorithms to reach the optimum. There were 50 trials performed per group. Unless otherwise stated, confidence levels for all tests are 95%.

Table 4.1: Steady state CCEA results on the LEADINGONES problem. Each value represents the mean number of evaluations needed to reach the optimum out of 50 trials. From the top left corner, proceeding clockwise, the tables represent data for decompositional biases created using two, four, eight and sixteen populations.

$r = k$	# Collaborators		
	1	2	3
c-best	8841.56	17546.52	25331.32
c-rand	8838.46	17621.88	25640.28

$k = 4$	# Collaborators		
	1	2	3
c-best	9355.16	18994.36	28061.20
c-rand	10636.90	19229.02	28769.28

$k = 8$	# Collaborators		
	1	2	3
c-best	10420.48	21178.52	31253.44
c-rand	13161.24	23657.76	34302.52

$k = 16$	# Collaborators		
	1	2	3
c-best	11930.08	23723.20	34587.76
c-rand	19458.44	30713.74	42862.28

Statistically, the relevant groups were compared with a t-test using the Bonferoni adjustment. In all cases choosing one collaborator is significantly better. This might at first be puzzling since there is clearly cross-population epistasis present when $k > 2$. However, note that a mutation that turns some bit to 1 in an individual in the first population will always result either a neutral or positive change in fitness, regardless of the contents of the other population. The reverse is also true. In addition, this fact is not true symmetrically for $k = 4$. The second and fourth populations will remain relatively unimportant to the first and third populations for some time during the evolution, since each LEADINGONES subproblem is solved left-to-right. This is essentially the same observation made by the formal analysis of the CC (1+1) EA, above.

Observe that by changing the number of populations, the mutation rate is effectively being increased (recall that the mutation is $1/l$, where l is the number of bits per individual in each population). Such issues may affect the empirical results, consequently I ran all population-oriented experiments discussed above with a constant $1/64$ mutation rate. The results (below) remain consistent with those reported above. Again, statistically speaking it is better to select only a single collaborator.

Recall that one observation about the LEADINGONES problem made during formal analysis earlier in this chapter was that there is nothing intrinsically contradictory in the interrelated bits of this problem. This can be

Table 4.2: Steady state CCEA results on the LEADINGONES problem with a constant mutation rate of 1/64 for all experimental groups. Each value represents the mean number of evaluations needed to reach the optimum out of 50 trials. From the top left corner, proceeding clockwise, the tables represent data for decompositional biases created using two, four, eight and sixteen populations.

$r = k$	# Collaborators		
	1	2	3
c-best	8428.28	17315.64	26535.94
c-rand	9103.48	17128.94	26677.74

$k = 4$	# Collaborators		
	1	2	3
c-best	12646.04	27122.52	37218.52
c-rand	13759.30	27517.72	39194.62

$k = 8$	# Collaborators		
	1	2	3
c-best	22654.56	42379.00	68231.68
c-rand	24298.46	45986.04	69805.16

$k = 16$	# Collaborators		
	1	2	3
c-best	40193.10	79460.12	119571.76
c-rand	43751.84	83424.06	124537.00

easily rectified by a simple change to the problem: scale the LEADINGONES part by s and subtract ONEMAX from the total, i.e., $g(x^{(i)}) = s \cdot \text{LEADINGONES}(x^{(i)}) - \text{ONEMAX}(x^{(i)}) + s$, and $f(x) = \sum_{i=1}^{k} g(x^{(i)})$. The entire function is translated up by s to prevent any negative values that will create a difficulty for the fitness proportionate selection method.

With this new function the right side will affect the search since bits to the right of the left-most zero bit will contribute to fitness, but there is some tension between individual bit contributions to fitness and those of their non-linear interactions since those right-side bits contribute *negatively*. This modification is a half-step towards pure contradiction since this tension is one directional. Take the string: "11 00 00 . . ." as an example. Flipping the fourth bit to a one will decrease the fitness slightly if the third bit remains 0, while flipping both the third and fourth bits will increase the fitness. However, the same is not true on the other side. Flipping the third bit while the fourth bit remains 0 will also increase fitness. So some of the interactions have this property of sign-dependent epistasis, while others will not. In addition, the linear effects of the bits are very muted compared to the non-linear effects due to the scaling issue.

This $s \cdot \text{LEADINGONES} - \text{ONEMAX}$ function is again concatenated using two pieces ($r = 2$), that is $f(x) = g\left(x^{(1)}\right) + g\left(x^{(2)}\right)$. Again, I applied the SS-CCEA algorithm using the 6 collaboration methods described above. The results for $k = 2$ through $k = 16$ in Table 4.3 show the average number of evaluations it took the algorithms to reach the optimum (50 trials each).

In addition to the data presented, I performed similar experiments at higher k values. In all cases there was again no statistical reason to choose another collaboration method other than the single best individual from the other populations. Not only does this increased decompositional bias not alter the collaboration methodology, it appears as though this problem becomes *easier* for the CCEA to solve, not harder, when using the c-best strategies. This turns out to be statistically significant only for the $k = 16$ and $k = 64$ cases where using the single-best strategy, for the $k \geq 16$ cases using 2-best, and the $k = 32$ and $k = 64$ cases using 3-best. Figure 4.3.2 below shows results for the c-best strategies as k is increased in $\{2, 4, 8, 16, 32, 64\}$.

So far, these experiments confirm what one sees in practice, namely that the simple collaboration method involving just the best individuals from each population is quite robust even when there is cross-population epistasis. However, what is still not clear is when it fails. To understand that better, I next focus on the the various forms of cross-population epistasis.

Table 4.3: Steady state CCEA results on the $s \cdot \text{LeadingOnes} - \text{OneMax}$ problem. Each value represents the mean number of evaluations needed to reach the optimum out of 50 trials. From the top left corner, proceeding clockwise, the tables represent data for decompositional biases created using two, four, eight and sixteen populations.

$r = k$	# Collaborators		
	1	2	3
c-best	15082.82	30814.68	45780.04
c-rand	16207.60	29958.68	46421.48

$k = 4$	# Collaborators		
	1	2	3
c-best	15233.84	30805.00	44637.04
c-rand	16213.30	29664.68	45027.10

$k = 8$	# Collaborators		
	1	2	3
c-best	14407.84	29190.36	42922.96
c-rand	16437.56	28516.20	46327.98

$k = 16$	# Collaborators		
	1	2	3
c-best	13996.32	28421.08	43566.16
c-rand	20674.84	34724.52	49562.24

4.3.3 Linkage Bias

While decompositional bias focusses on potential mismatches between a problem's "natural" decomposition in terms of separability and the number of CCEA populations used, there may still be cross-population non-linearities when the design engineer decomposes the problem into the same number of components as there are separable pieces of the problem. This is because it is possible to assign lower-order components with such non-linearities to different populations. Breaking up tightly linked bits can result in significant cross-population epistasis. Indeed, as I will show at the end of this chapter, this type of non-linearity can become pathological to CCEAs in some cases. In general, the degree to which linked bits in a piece are assigned to the same population for the purposes of representation can be thought of as *linkage bias*.

Once again it is easy to construct a method for controlling this bias: define a mask over the entire bit string that specifies to which population a given bit belongs, $\mathcal{M} \in \{1, 2, \ldots, k\}^n$. Note that in the case of these mask definitions, the superscript suggests repetition, and not an exponent. For problems like those in the previous section involving a series of r concatenated non-decomposable s-bit blocks and assuming for the moment that $r = k$ and $s = l$, a mask that corresponds to the most biased linkage (i.e. is more closely aligned with the real problem, or *matches* the problem well) is $\mathcal{M}_s = 1^l 2^l \ldots k^l$. Coming up with a highly pathological mask is very problem dependent, but a mask that will turn out to be commonly quite bad is $\mathcal{M}_h = (123\ldots k)^l$. Here every bit in a block is distributed to every population, resulting in the likelihood of a high degree of cross-population epistasis for most common problems.

Again, increases in the amount of cross-population epistasis in this case may affect a problem's difficulty to solve with a CCEA. By applying different types of masks, which distribute different pieces of the blocks of the problem to different degrees, I can explore the affect that varying degrees of linkage bias have on collaboration methods.

Consider again the $s \cdot \text{LeadingOnes} - \text{OneMax}$ problem, assuming $r = k = 2$. Using the $\mathcal{M}_s = 11\ldots1\ 22\ldots2$ mask produces the same decomposition already discussed, where there is no cross-population epistasis, while the mask $\mathcal{M}_h = 1212\ldots12\ 2121\ldots21$ creates a situation with very strong cross-population epistasis. Once more I apply the SS-CCEA problem, with the 6 collaboration methods described above; however, this time I compare the effects that these two masks had on the choice of collaboration methods, rather than the number of populations used. The results are presented in Table 4.4.

Differences between the means for c-best and c-random groups for $\mathcal{M}_h$ are significant for one and two

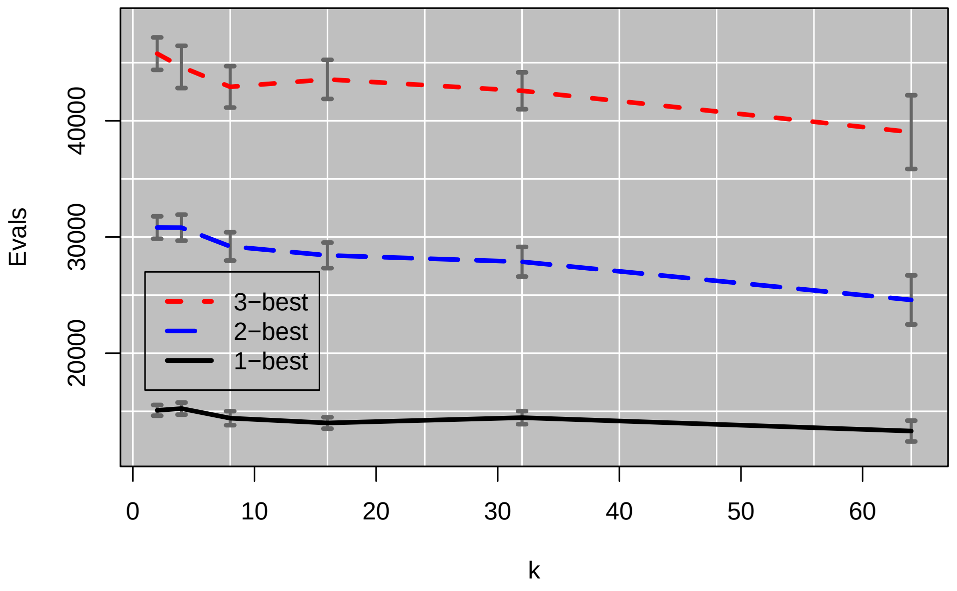


Figure 4.7: Results for the SS-CCEA on the $s \cdot$ LEADINGONES − ONEMAX problem for increasing values of k. The bottom curve (solid) illustrates results for the single-best collaboration strategy, the middle curve (long dashed) represents results for the 2-best, and the top curve (short dashed) represents results for the 3-best. The x-axis shows the size of k, and the the y-axis shows the number of function evaluations until the global optimum was reached. The vertical wings show the 95% confidence interval of 50 independent trials; the plotted point shows the mean.

collaborators, but not for three. There are no statistically significant differences between these groups (for the same number of collaborators) for the simpler linkage bias.

Once again simply distributing the epistatic linkages across the population boundaries is insufficient to require that a more complicated collaboration method be used. This may seem surprising at first, but note that, for this particular problem, introducing such masks does not change the *type* of cross-population epistasis, only its degree. Moreover, although not germane to our question, it is interesting to note that in this particular case increasing the mixing seems to *improve* performance versus the $\mathcal{M}_s$ mask (this is significant for all but the 3-random case).

What remains to be shown is what the effect of the most difficult form of cross-population epistasis has on collaboration selection. When neither the sign nor the magnitude of the interaction can be reliably predicted, will it become necessary to select collaborators differently? Intuition seems to suggest that it will. If no prediction can be made about the higher order blocks from the lower order blocks, then a more sophisticated method of sampling the interaction space should be needed. What I need to do now is construct a problem with just such a challenge.

Fortunately, the $\text{CLOB}_{b,k}$ problem mentioned previously has exactly this property. Note that this problem is already one constructed by concatenation of functions over smaller, inseparable pieces. Because this problem is already quite difficult when $b = 2$ and I am looking at linkage bias, rather than decompositional bias, here the focus is on the specific function $\text{CLOB}_{2,2}$. This offers the simplest such problem still more complicated than the $s \cdot$ LEADINGONES − ONEMAX. As it turns out, it is so difficult under the $\mathcal{M}_h$ mask that the algorithms did not find the global optimum with $n = 128$. The reason for this should be clear by the end of this chapter. At any rate, as a consequence Table 4.5 represents the mean fitness values obtained after a fixed budget of 100,000 evaluations. The reader should be careful to note that in this case higher values are better.

Table 4.4: Steady state CCEA results on the $s \cdot \textsc{LeadingOnes} - \textsc{OneMax}$ problem for different linkage biases. Each value represents the mean number of evaluations needed to reach the optimum out of 50 trials. The left table represents a linkage bias that uses the $\mathcal{M}_s$ mask, while the right uses the $\mathcal{M}_h$ mask.

$\mathcal{M}_s$	# Collaborators		
	1	2	3
c-best	15472.42	30393.32	46064.92
c-rand	15669.34	30830.90	45401.42

$\mathcal{M}_h$	# Collaborators		
	1	2	3
c-best	15558.96	29963.64	45702.88
c-rand	17417.48	32920.20	47527.80

Table 4.5: Steady state CCEA results of the $\text{CLOB}_{2,2}$ problem. Each value represents the mean value obtained after 100,000 function evaluations out of 50 trials. The left table represents a linkage bias that uses the $\mathcal{M}_s$ mask, while the right uses the $\mathcal{M}_h$ mask.

$\mathcal{M}_s$	# Collaborators		
	1	2	3
c-best	1811.22	1140.4	1002.78
c-rand	1708.34	1167.63	1022.57

$\mathcal{M}_h$	# Collaborators		
	1	2	3
c-best	379.1	477.06	499.38
c-rand	782.65	788.74	810.5

Now we see exactly the reverse situation as we did before. Although it is clearly better to select a single collaborator when there are no cross-population epistatic linkages of this type, as soon as those linkages are spread across populations a more complex collaboration mechanism is required. In the latter case, increasing the number of collaborators does in fact result in statistically improved performance. Specifically, using one collaborator is significantly worse than using two or three. This statistical result was obtained using Tukey-Means at a 95% confidence interval and was confirmed using Fisher LSD. The same result was obtained using pairwise t-tests and the Bonferoni adjustment when run for 100 trials (not reported). It was also the case that picking collaborators randomly was statistically superior to picking them greedily when there was cross-population epistasis.

Though the property of inseparability has a role to play in terms of informing us of how one selects collaborators, that role is not a simple one. The formal analysis revealed that the mere presence of cross-population epistasis is insufficient to guarantee either that the CCEA will be expected to perform better or worse than the EA. The above empirical analysis indicates that the simple presence of cross-population epistasis is also not a reason to prefer more complicate collaboration methods. In fact, it appears to be the *type* of non-linearity that is relevant. More sophisticated methods are needed when neither the sign nor the magnitude of the fitness change can be predicted from changes in the lower order pieces. This makes sense: if collaboration is about sampling the interaction space, there needs to be a reason to sample. The biggest reason to sample is when lower order pieces can be contradictory to the result, thus requiring the algorithm to discover more sophisticated information about these linkages. I will return to this idea in a bit, but first let's turn our attention to properties of the collaboration choice itself.

4.3.4 Selecting Collaborators

In the previous subsection, I identified the salient properties in the problem that result in the need for more sophisticated methods for collaboration. In this subsection, I will use the $\text{CLOB}_{2,2}$ problem examples with both the simple and hard masks to discuss the other side of this coin: what kinds of choices can be made when selecting collaborators, and what results should one expect from these choices? Recall that there are essentially three attributes to consider for collaborator selection: sample size, selective pressure, and credit assignment.

The first case is simple since the sample size can be controlled simply by controlling the number of collaborators selected. As above, I will use c to denote the size of the collaborator sample. Of course, the larger this value is, the more objective evaluations will be required per individual. It should be clear that increasing the sample size and the number of populations together can have a combinatorial effect on these function evaluations. So, in general, it would seem that the fewer collaborators *we need* to obtain "good" results, the better.

The second case requires a slightly more complicated mechanism. In this case, a simple tournament draw method can control the degree of selective bias of collaborators. That is, the algorithm selects q collaborators at random with replacement, and chooses the one with the highest fitness from its previous evaluation. This does not increase the number of function evaluations performed since the algorithm simply uses the results from previous evaluation. The net effect is that the selection pressure can be increased from purely random ($q = 1$) to elitist ($q :\approx p \lg p$, where p is the population size).

When multiple collaborators are evaluated for the purposes of fitness (when $c > 1$), there is still the open question of how one assigns a single fitness score from the results of multiple objective function evaluations, collaborator credit assignment. Here only three possible credit assignment methods are considered: *optimistic*, *hedge*, and *pessimistic*. These correspond respectively to using the max, mean, and min of the c available objective function results.

Let's consider collaborator credit assignment first, since the results of credit assignment are perhaps the most obvious. I applied the two population SS-CCEA to the $\mathrm{CLOB}_{2,2}$ problem under each of the two masks. Again the population size is 10, bit-flip mutation at a rate of $1/l$ was used, and the total length of the bit string was 64 bits. I ran this algorithm 50 times against these two problems using all three credit assignment methods varying $c \in \{2,3,4,5\}$. The table below displays the mean best fitness found after 100,000 function evaluations is computed. Though the hedge and pessimistic methods may not *use* the objective value for the purposes of fitness assignment, any value computed *during* the credit assignment process (before a score is selected to be assigned) is available for consideration of the best-so-far information reported. This alleviates any unfair advantage the optimistic method may have in this respect. Not surprisingly, the result is a clear statistical advantage to the optimistic approach. In the case of the simple mask, none of the three approaches is statistically superior to either of the others for any collaborator sample size. For $\mathcal{M}_h$, the optimistic case is statistically better than both the hedge and pessimistic methods.

Table 4.6: Collaboration credit assignment results for the steady state CCEA applied to the $\mathrm{CLOB}_{2,2}$ problem using two different masks. Each value represents the mean best-so-far of 100,000 objective function evaluations out of 50 trials. The top table represents a linkage bias that uses the $\mathcal{M}_s$ mask, while the bottom uses the $\mathcal{M}_h$ mask.

$\mathcal{M}_s$	# Collaborators			
	2	3	4	5
Optimistic	1210.50	984.64	919.04	823.56
Hedge	1216.66	966.14	879.36	817.32
Pessimistic	1226.56	964.96	840.96	828.54

$\mathcal{M}_h$	# Collaborators			
	2	3	4	5
Optimistic	465.28	966.14	595.88	537.74
Hedge	460.32	504.96	507.44	526.04
Pessimistic	361.56	406.56	397.70	379.50

Let's now turn our attention to the remaining two parameters: sample size and selective bias. Again, the SS-CCEA was applied to these two problems. This time a total of 50 experimental groups were used for each of the two problems: $c \in \{1,2,3,4,5\}$ and $q \in \{1,2,3,4,5\}$. Figure 4.3.4 below illustrates the results of 50 trials of each group for the $\mathrm{CLOB}_{2,2}$ problem using the simple mask, while Figure 4.3.4 below illustrates the results of 50 trials of each group for the $\mathrm{CLOB}_{2,2}$ problem using the hard mask.

The results of these experiments are a bit more interesting. Barring a couple of exceptions, there is no statistical advantage to biasing the selection process for the problem with the simple mask. The exceptions are that the $q = 2$ and $q = 4$ groups differ for the two collaborator case, but not for any other collaborator sample size. The reverse, however, is not true. Clearly the sample size hinders performance in all but a few cases. There is no statistical difference between $c = 3$ and $c = 4$ for the $q = 1,2,$ and 3 groups, and there is no statistical difference between $c = 4$ and $c = 5$ for any of the groups.

Likewise, when using the more complex mask, biasing did not appear to affect the results. In this case, no q value was superior to another at a particular c value. Again the collaborator samples size is crucial, but this time its service is reversed: increasing the sample size leads to better performance.

What is the message? When there is significant contradictory cross-population epistasis, practitioners should consider using a more sophisticated collaboration method. In the case of static function optimization, using an optimistic credit assignment method is typically a good choice. Finally, between increasing the sample size and biasing the sample, the former is more effective. This is consistent with observations made from existing research (Wiegand, Liles, and De Jong 2001; Bull 1997).

4.4 Difficulties with CCEAs

The work leading up to this point has shown that CCEAs can sometimes outperform traditional EAs, even when cross-population non-linearities are present. Moreover, it has also been shown that the reverse is true as well: there are inseparable problems for which the EA outperforms the CCEA. The conclusion to draw here is not that the CCEA is better or worse than the EA, but that there are properties of the problem that lend themselves toward CCEAs, and those that lend themselves to EAs. This is not an altogether surprising conclusion, of course, especially given the No Free Lunch theorem.

However, the crucial piece of (perhaps counter intuitive) understanding is this: the relevant property controlling CCEA performance on static optimization problems is something more complicated than merely the presence or absence of separability. The non-linear relationships between components represented in populations may or may not create difficulties for coevolutionary algorithms. The issue is clearly more complicated than this. There are some canonical problem properties that tend to lead to difficulties for coevolutionary algorithms, but the multi-population symmetric algorithms discussed in this thesis are carefully designed to avoid these difficulties. For example, intransitive relationships in the rewards are not possible in the class of CCEAs under investigation. If not the presence of cross-population epistasis (or, even, the *degree* of such), and if not the more recognizable coevolutionary stumbling blocks, what then are the relevant properties?

There are at least two important properties that complicate CCEA search: contradictory non-linear effects between populations and consensus in the joint distributions of strategy rewards. Both of these properties often cause CCEAs to fail to find the global optimum of an objective function. The subsections below will discuss both of these, but the first is emphasized here since the second is addressed much more extensively in Chapter 5.

4.4.1 Contradictory Cross-Population Epistasis

The idea that there are different *types* of epistasis is not new. Reeves (2002) shows that the original study of epistasis by Davidor (1990) is flawed, in part, because of his assumption that it is only existence of epistasis that matters for a traditional GA, and not the *type* of epistasis. Indeed, as for GA hardness, the type of epistasis (in this case, cross-population epistasis) is the important property. The *direction* (sign) of the non-linear contributions with respect to the linear contributions is important. Here what is meant by *contradictory cross-population*

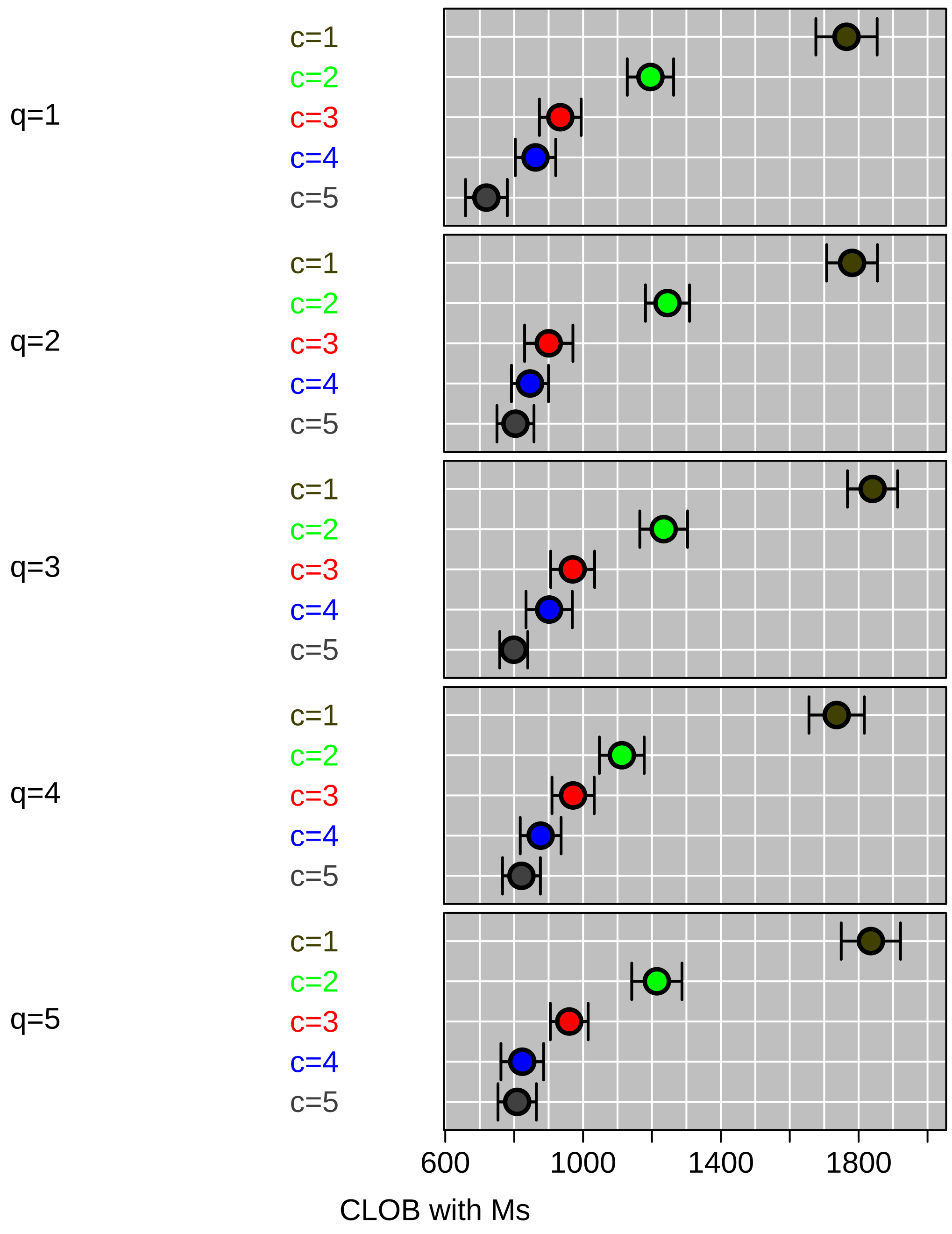


Figure 4.8: Collaboration selection results for the SS-CCEA on the CLOB$_{2,4}$ problem using the $\mathcal{M}_s$ mask for linkage bias. The *x*-axis shows the best-so-far of 100,000 objective function evaluations. The panels (from top to bottom) represent increasing selective pressure (q), while the items in each panel represent increasing sample size (c). The points are plotted at the mean of 50 trials, while the horizontal wings show the 95% confidence intervals.

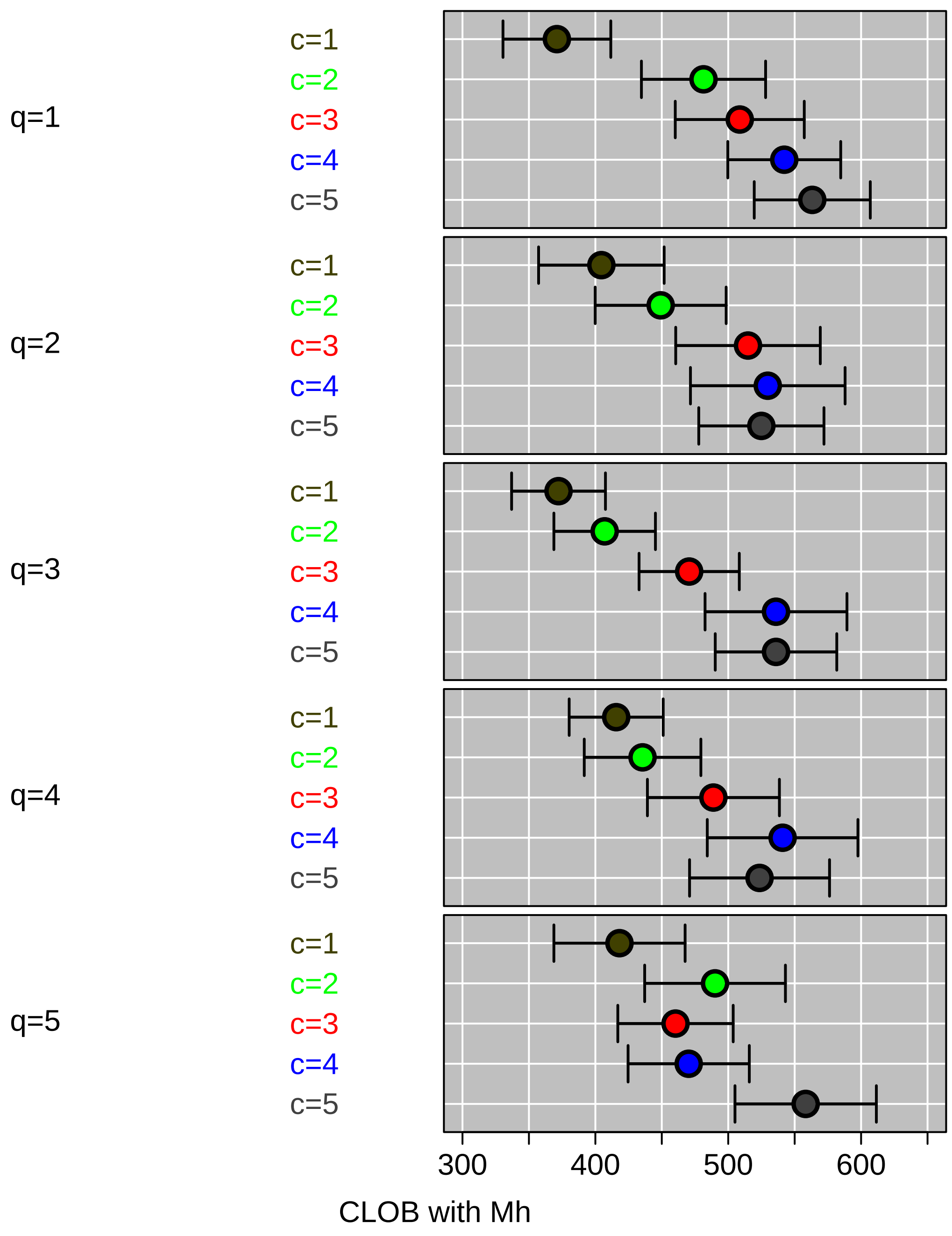


Figure 4.9: Collaboration selection results for the SS-CCEA on the CLOB$_{2,4}$ problem using the $\mathcal{M}_h$ mask for linkage bias. The *x*-axis shows the best-so-far of 100,000 objective function evaluations. The panels (from top to bottom) represent increasing selective pressure (q), while the items in each panel represent increasing sample size (c). The points are plotted at the mean of 50 trials, while the horizontal wings show the 95% confidence intervals.

epistasis is the situation in which the linear effects of lower order components existing in different populations are opposite in magnitude *and* direction of the higher order blocks. This is a type of cross-population deception where the algorithm is tricked into believing that the lower order components accurately predict the total function value, when in fact they are misleading.

This is particularly damaging in a coevolutionary process that locks off all but the current component while conducting a projected search. It is not hard to see that the same partitioning process that assists the CCEA in gaining advantage against the EA in certain situations can become its Achilles heal in the presence of this deception. Consider the CC (1+1) EA again, now working on a TRAP function, presented in Definition 10 on page 34. This function is unsolvable by the CC (1+1) EA presented at the start of this chapter. Consider the following proof.

Theorem 3. *Let* $\textsc{Trap}\colon \{0,1\}^n \to \mathbb{R}$*, be decomposed into* $k \geq 4$ *equal sized components of length* $l \geq 2$*, such that* $n = kl$*. The sequential CC (1+1) EA will fail to converge to the global optimum of* TRAP *with probability* $1 - 2^{-\Omega(n)}$.

Proof. Define the term *solved component* to mean a component that is the all one string, 1^l, and the term *unsolved component* to mean a component that contains at least one 0.

The proof consists of two basic parts. First, it is proved that with a probability exponentially approaching 1, there are at least two subpopulations of the sequential CC (1+1) EA whose individual is an unsolved component. Next it is shown that, given there are at least two unsolved components after initialization, the mechanism of the algorithm itself will prevent it from accepting mutations that will lead it to the global optimum.

For the first part of the proof, observe that the probability that a given population's individual contains the all one string after initialization equals $\frac{1}{2^l}$, so the expected number of such individuals in k subpopulations equals $\frac{k}{2^l}$. Chernoff bounds gives the probability that at most half of the k subpopulations are solved after initialization is at most

$$
\begin{aligned}
& \left[\frac{e^{2^{l-1}-1}}{(2^{l-1})^{2^{l-1}}}\right]^{k/2^l} \\
= & \quad \frac{1}{e^{k/2^l}}\left(\frac{e}{2^{l-1}}\right)^{k/2^l \cdot 2^{l-1}} \\
= & \quad \frac{1}{e^{k/2^l}}\left(\frac{2e}{2^l}\right)^{k/2} \\
= & \quad 2^{k/2-kl/2} \cdot e^{k/2-k/2^l} \\
= & \quad \frac{1}{2^{\Omega(kl)}} = 2^{-\Omega(n)}
\end{aligned}
$$

So the probability that at least half of the k populations are unsolved is $1 - 2^{-\Omega(n)}$. Thus, given $k \geq 4$, there are at least two unsolved components with probability exponentially approaching 1.

For the second component of the proof, let us first suppose that all the subpopulations contained the all one string except for two of them, one that is the current active subpopulation under consideration by the algorithm, population a, and another that has yet to be considered during the current round, population b. Suppose that the necessary mutations are performed in $x^{(a)}$ such that the offspring, $y^{(a)}$ is the all one string. The offspring $y^{(a)}$ can be accepted if and only if the following holds.

$$\textsc{Trap}(x_{t+1}^{(1)} \cdots y^{(a)} \cdots x_t^{(b)} \cdots x_t^{(k)}) \geq \textsc{Trap}(x_{t+1}^{(1)} \cdots x_t^{(a)} \cdots x_t^{(b)} \cdots x_t^{(k)})$$

However, this cannot be true since $x_t^{(b)}$ does not contain the all one string. Therefore the offspring will not be accepted. Since the parent, $x_t^{(a)}$ was not the all one string, the same event will be true symmetrically for $x_t^{(b)}$.

Given that $k \geq 4$, there cannot be fewer than two unsolved components. Having more than two unsolved components cannot resolve the problem, since an offspring that contains more ones than the parent will have a lower fitness than the parent if at least one other component is unsolved. □

Is this result surprising? Perhaps it is a function of the fact that the algorithm processes the populations sequentially, rather than updating all the populations simultaneously and in parallel to advance to the next round. Unfortunately this is not the case, as the following theorem and proof demonstrate.

Theorem 4. *Let* $\textsc{Trap} : \{0,1\}^n \to \mathbb{R}$*, be decomposed into* $k \geq 4$ *equal sized components of length* $l \geq 2$*, such that* $n = kl$*. The parallel CC (1+1) EA will fail to find the global optimum of* $\textsc{Trap}$ *with probability* $1 - 2^{-\Omega(n)}$.

Proof. Recall that the term *solved component* refers to the the all one string, 1^l.

Again, the proof consists of two basic parts. The first part is obtained from the proof to Theorem 3, since the two algorithms do not differ with respect to initialization. With a probability exponentially approaching 1, there are at least two subpopulations of the parallel CC (1+1) EA whose individuals are unsolved components. The second part is also similar: it must be shown that given there are at least two unsolved components after initialization, the mechanism of the algorithm itself will prevent it from accepting mutations that will lead it to the global optimum.

To see this, first suppose that all the subpopulations contained the all one string except for two of them, one that is the current active subpopulation under consideration by the algorithm, population a, and another that has yet to be considered during the current round, population b. Suppose that the necessary mutations are performed in $x_t^{(a)}$ such that the offspring, $y^{(a)}$ is the all one string. The offspring $y^{(a)}$ can be accepted if and only if $\textsc{Trap}(x_t^{(1)} \cdots y^{(a)} \cdots x_t^{(b)} \cdots x_t^{(k)}) \geq \textsc{Trap}(x_t^{(1)} \cdots x_t^{(a)} \cdots x_t^{(b)} \cdots x^{(k)})_t$, which cannot be true, since $x_t^{(b)}$ does not contain the all one string. Therefore the offspring will not be accepted. Since the parent, $x_t^{(a)}$ was not the all one string, the same event will be true symmetrically for $x_t^{(b)}$. Given that $k \geq 4$, there cannot be fewer than two unsolved components. Again, having more than two unsolved components cannot resolve the problem, for the same reason as in the last proof. □

This problem cannot be solved by the CC (1+1) EA even when the update mechanism is performed in parallel, no matter how much time the algorithm waits. The likelihood of finding the solution on initialization is $1 - 2^{-\Omega(n)}$, but even an infinite number of steps will not be sufficient to discover the solution in the event that it is not discovered during initialization. The reason is that the algorithm is misled by the problem to solve the $-\textsc{OneMax}$ portion, leading to the bit string of all zeros; however, no single n-bit mutation is possible due to the division (irrespective of the updating mechanism). The problem can be made less difficult by aggregating many smaller trap functions, producing a problem that can be solved by an EA in a tunable amount of time (depending on the size of the traps), but *still* can never be solved by the CCEA assuming there is still a division that splits at least one trap between two populations.

Is this result attributable to the fact that only a single parent individual exists in the population? Again, unfortunately this is not the reason. Many larger population approaches (including the SS-CCEA) will not resolve the problem even when the populations are very large and arbitrarily complex collaboration mechanisms are used. The populations will quickly become flooded by the all zero string. Maintaining enough diversity to preserve at least one all one string component in even one population would require a population size is $\Omega(2^n)$. Looking back at earlier parts of the chapter, this is exactly why the $\textsc{Clob}_{2,2}$ problem using the $\mathcal{M}_h$ mask could not be solved by the SS-CCEA, and why neither the sequential nor parallel variant of the CC (1+1) EA can solve the $\textsc{SepFun}$ problem.

The optimization problem need not be so difficult to see this pathology. Consider the following $r = 1$ problem that nevertheless divides the bit string into two equal substrings of 8 bits apiece (for a total of 16 bits).

$$\begin{aligned} f(x) = \quad & n \cdot \left[\textsc{LeadingOnes}\left(x^{(1)}\right) \cdot \textsc{LeadingOnes}\left(x^{(2)}\right)\right] - \\ & \textsc{LeadingOnes}\left(x^{(1)}\right) - \textsc{LeadingOnes}\left(x^{(2)}\right) + n \end{aligned}$$

I applied the same two population SS-CCEA used earlier in the chapter on this problem, ranging $c \in \{1,2,3,4,5\}$ and running the algorithm for 100,000 function evaluations. The global optimum was never found out of 50 trials (the reader should note that 65,536 evaluations would be needed to find the optimum under full enumeration). In fact, as should be clear from the above analysis, the highest value ever found by the SS-CCEA was 16.0. The SS-EA found the optimum in, on average, 208.46 evaluations.

4.4.2 Consensus in Joint Distributions

The second difficulty that challenges the CCEA is much more subtle. If one considers the rewards of the payoff function for fitness as a mere projection in the search space, it becomes clear that the collaborators, or samples of the interactions, are samples from a joint distribution across all conjoining strategies.

Consider that many potential collaborators are unlikely to be represented in the other populations, much less be selected for collaboration. Given this, it is possible that such a sample may misleading. The bias in the alternate population indicates a possible bias in the sample. Since high rewards are likely to draw populations, those problems for which there are local peaks having large joint distribution values are likely to pull populations away from peaks with smaller joint distribution values, even when the peak is much larger. This is a form of "rule of the mass" that misdirects search into areas that are suboptimal, because the populations themselves prefer the consensus. In other words, CCEAs working on problems with broad, but suboptimal peaks and narrow peaks for the global optima, will tend to converge the the broad suboptimal peaks. This effect is the predominant subject of Chapter 5.

Obviously such a fact is often true of many traditional EAs, as well; however, in the case of the CCEA the pathology is more endemic to the structure of the algorithm itself. Consider that even a perfect local optimizer working on an incorrect component projection of the problem will suboptimally converge. The CCEA is directed, not just by the problem itself, but by the interacting dynamics between the populations working on that problem. As larger joint distributions attract one population, that population becomes a degenerate representation for the others. This idea will be clarified formally in the next chapter since the idea ultimately stems from the fact that the CCEA is not really meant to optimize the external static objective functions as described so far.

Chapter 5

Optimization versus Balance

One key property with all coevolutionary systems is that of *balance*. At least two different notions of balance are important to CCEAs. First, the systems tend to continue to make progress only as long as a certain degree of balance in evolutionary change is maintained dynamically during the run of the algorithm, called *adaptive evolutionary balance*. The second, more important form of balance is called *robust resting balance*. Trajectories describing dynamic behaviors of CCEAs tend to come to rest in parts of the space that offer a type of game-theoretic balance between populations: the most important characteristic for stability is that both populations consist of individuals representing strategies that can do no worse when collaborating with the individuals from the other populations.

This latter notion of balance is a familiar one in game-theory: it is exactly the idea of Nash equilibria (Nash 1950). Unfortunately it turns out that in many cases the most attractive Nash equilibria for coevolutionary systems are not ones that correspond with ideal partnership (i.e., the specific collaboration that results in the optimal fitness) in any objective sense. This chapter makes use of a familiar game-theoretic modeling technique for coevolution in order to explore these notions of "balance" more explicitly. In particular, the chapter looks at the traditional goals of optimization applications as they are often encoded into CCEAs and examines how well these goals correspond with the CCEA propensity towards balance. I begin by first offering the assertion that naively applying CCEAs to static optimization problems may constitute a fundamental failure to appreciate the true function of these algorithms. The two sections that follow first defines, then analyzes the evolutionary game-theoretic model in order to justify this assertion. In the final section, I provide empirical investigations based on the theoretical results demonstrating real consequences for CCEAs in applied settings.

5.1 CEAs are Not Static Optimizers of Ideal Partnership

A cooperative coevolutionary algorithm can be modeled as a dynamical system. Like many dynamical systems, trajectories describing system behaviors over time tend to move toward, and settle in, stable fixed points—points that often have certain (sometimes predictable) properties, such as the afore mentioned game-theoretic balance. To some extent the question of whether or not cooperatively coevolutionary systems optimize is semantic: in fact, in a sense CCEAs *do* optimize. It is simply a question of *what* they optimize.

In the previous chapter, the reader was asked to assume that CCEAs are well-suited to static optimization problems that are straightforwardly encoded into a representation in more or less the same way that traditional EA static optimization problems are encoded. In such a situation there is an implied optimization goal: find the arguments that correspond to the optimal objective value (see Definition 7 on page 29, for example). But what is the *meaning* of this argument in a cooperative coevolutionary setting?

Let us define the term *ideal partnership* (or *ideal collaboration*) to mean a particular set of all component strategies of a problem as yielding the highest objective reward. In other words: assuming each population offers a particular individual to be evaluated, if the collaboration yields the highest possible reward of any collaboration, we consider it ideal. Consequently, such collaborations would correspond with the global maximum of the problem, at least insomuch as it is defined by the encoding scheme so far assumed. To believe that CCEAs are "well-suited" for optimization problems represented in this way, is to believe that they are dynamically predisposed to move toward (and rest in) spots in the search space that correspond with the ideal partnership.

Unfortunately, this is not the case. Importantly, its failure to be so driven results not from stochastic error, or being mislead by sampling of the search space (as is often the case with traditional EAs), but by the CCEA's very nature to move toward parts of the space that are not necessarily considered "optimal" in the above sense. In other words, even with perfect information, infinite population sizes, and no probabilistic noise, CCEAs can be expected to settle in suboptimal areas of the space. Alternatively, it has been shown in many places, traditional EAs do not suffer from this affliction (Reeves and Rowe 2002; Schmitt 2001; Vose 1999).

In the previous chapter, I discussed two properties of problem domains that can become stumbling blocks for CCEAS: contradictory cross-population epistasis and consensus in joint distributions of the reward. Both of these properties can lead to an optimization failure. However, knowing that these problem properties are possible, and they *might* lead to unfortunate *results*, does not help us understand how certain behaviors observed in CEAs relate to these problems. What are we looking for in the dynamics of a CEA? What is good and what is bad? Indeed, the goal of this chapter is to uncover some of these *dynamic* properties of the CCEA.

I begin by providing a brief overview of some dynamical behaviors associated with coevolutionary algorithms. I attempt to classify these behaviors in a tangible and appropriate way. After the analysis presented in this chapter, I will offer specific empirical examples demonstrating two of these pathological behaviors in CCEAs. The point of this section, however, is to show some of the symptoms of common afflictions of CEAs such that the reader will have some perspective for the theoretical analysis to follow, which attempts to explain two specific pathologies of cooperative coevolution in greater detail. These pathologies (*loss of gradient* and *relative overgeneralization*) are, in fact, outward symptoms of the more fundamental problem: CCEAs are not static optimizers of ideal partnership.

5.1.1 Overview of CEA Dynamics & Pathologies

This section provides an overview of the types of dynamics commonly observed in coevolutionary algorithms. Some of these dynamics have been mentioned in passing in the introduction. Frequently they bring to mind certain qualitative judgments: fecund (good) behaviors and pathological (bad) behaviors. Alternatively, other behaviors are merely descriptive of certain characteristics observed of CEAs during run time. A more directed, specific discussion is needed. To be clear, the discussion is broken into two categories: fecund behaviors and pathological behaviors. To suit the interests of the viewpoint of this thesis, some of the early terms used to describe various behaviors are altered for clarity and context.

Fecund Behaviors

What promise coevolution suggests! In the case of competitive coevolution, practitioners are attracted by its potential for inherent adaptiveness. It offers the opportunity to adaptively focussing a search on only those portions of a space that are important—of driving populations towards increasingly better results by the simple mechanisms of co-adaptation between populations. Indeed, sometimes CEAs work in exactly this way, a behavior typically referred to as an *arms race*. More explicitly,

Definition 19. ***Arms race*** *– A behavior in a coevolutionary system in which changes in one group or population results in reciprocal, co-adaptive changes in the other groups or populations, and by iterative application of this process the system produces increasingly better performing individuals by an external measure.*

To take a competitive example, consider two populations evolving game-playing strategies against the opposing population for some arbitrary game. In an arms race, when players in one population learn some new tactic to gain an advantage over players in the second population, the second is forced to respond and adapt. If this innovation–response cycle continues indefinitely, there seems to be the opportunity for limitless progress for both players.

Of course, the very term *arms race* implies a competitive system; however, a similar phenomena can be observed in cooperative models. The term is often used in cooperative domains, as well, though perhaps it would be more aptly (and generally) considered a type of balanced evolutionary change. Indeed, this is exactly the property mentioned above: *adaptive evolutionary balance*. Preserving such a balance in a cooperative CEA is akin to preserving an arms race in a competitive domain: as long as the balance is maintained, objective progress can continue.

Pathological Behaviors

Though productive behaviors from coevolutionary systems are behaviors for which we, as users of such systems, most desire, they are not the only behaviors, nor even the most common ones. Many coevolutionary behaviors lead to poor or middling external performance such as those described at the end of the last chapter. There are at least three common behaviors that are widely considered pathological in this way: *loss of gradient*, *relative overgeneralization*, and *mediocre objective stasis*. While all three can occur in competitive or cooperative algorithms, some are more problematic for one than for the other.

Perhaps the simplest to describe is *loss of gradient*. This term refers to the event that the search gradient suddenly becomes too steep to climb while a run is in progress.

Definition 20. ***Loss of gradient*** *– The coevolutionary behavior that occurs when one population or group reaches a state such that other groups and populations lose necessary relative fitness diversity from which to continue meaningful progress.*

A classic example of a lack of gradient between competitive coevolutionary opponents is the situation were a chess Grand Master plays a small child. If the child receives no information other than the outcome of the game, she has almost no means of learning how to improve her game. A “loss” of gradient can occur in this competitive setting when one population suddenly achieves a level so superior to the other, that simply nothing can be learned by either population by competing.

Loss of gradient is not strictly a competitive CEA problem, though. In a more general setting including cooperative CEAs, the same behavior can be seen as simply a loss of fitness diversity in one or both populations with respect to the other. That is: one or more populations suddenly loses diversity and the remaining population(s) are reduced to searching the projected space created by this diversity loss.

A different pathology can be seen in *relative overgeneralization*: the subjective nature of coevolution drives populations towards areas of the search space that are very general relative to the performance of other similar strategies. As a result, trajectories tend toward overly general solutions that may not be in any way optimal. Indeed, for cooperative algorithms, the result of such behaviors is *robust resting balance*.

Definition 21. ***relative overgeneralization*** *– The coevolutionary behavior that occurs when populations in the system are attracted towards areas of the space in which there are many strategies that perform well relative to the interacting partner.*

Again, this behavior can be observed in both competitive and cooperative algorithms, alike; however, stated as it is above, it is more an issue for cooperative algorithms than for competitive ones. Historically, papers discussing competitive algorithms have more often referred to such behaviors more generally as “relativism”. This alternate term reflects behaviors that result from situations in which the adaptive changes in a coevolutionary system become disconnected from some absolute measure and may be driven to parts of the space that are undesired (Watson and Pollack 2001). I do not use this definition, since it describes some sets of behaviors that can be categorized in “relative overgeneralization” group, and other sets that can be categorized in the next pathology (mediocre objective stasis). Such a division between these two groups is important for categorical purposes with respect to the underlying algorithms (cooperative versus competitive).

The final pathology is one that is primarily an issue for competitive algorithms: *mediocre objective stasis*. In such cases objective progress seems to have stalled while there is still apparent subjective change going on. Since the multi-population, symmetric CCEA has such an intimate connection between subjective and objective measures, this pathology is particularly unlikely, if not impossible.

Definition 22. ***mediocre objective stasis*** *– The coevolutionary behavior that occurs when there is no apparent progress according to some reasonable objective measure, despite continued adaptive subjective steps in the interacting individuals and population(s).*

The term "mediocre objective stasis" corresponds, in part, with the more common term "mediocre stability" as well as some behaviors alternatively classified as "relativistic" (see above). I have altered the terminology for clarity since it is possible that such behaviors are *not* stable in the dynamical systems sense. The term specifically refers to a stalling in the objective measure, not necessarily in domain space (Watson and Pollack 2001; Ficici and Pollack 1998). I also include in this definition the "focussing problem" described by Watson and Pollack (2001), in which competitive coevolution fails to find general solutions as a result of alternating between two or more specialized exploitation of particular weaknesses.

5.2 Modeling CCEAs with EGT

In order to get a better understanding of the behaviors of cooperative coevolution, a dynamical systems model was constructed. It is a well-known model, which that exploits the game-theoretic properties of coevolution: an Evolutionary Game Theoretic model. This section will describe the model in detail, beginning with a short discussion about notation.

5.2.1 Notation

Throughout this chapter the following notional conventions will be used. Lower case letters will be used to represent real numbers and functions. Vectors will be lower case letters denoted as $\vec{x}$, while capital letters will be used to denote sets and matrices. Euclidean spaces will be denoted $\mathbb{R}^n$, where n is a positive integer and indicates the dimensionality of the space. Given a set of connected points in Euclidean space, X, the *interior* of it is denoted as $\text{int}(X)$. The boundary of X is denoted $\text{bnd}(X)$.

The n dimensional sub-Euclidean space given by $\Delta^n := \{\vec{x} : x_i \in [0,1] \text{ and } \sum_{i=1}^{n} x_i = 1\}$ is called the *unit simplex*. The Cartesian product of two simplexes is denoted $\Delta^n \times \Delta^m$. I represent infinite sized populations by describing ratios of genotypes in the total populations using a point in the unit simplex. A point in $\Delta^n \times \Delta^m$ describes the state of a two-population CEA.

In simpler models, omitting genetic operators entirely, no particular genotypic representation is implied or assumed by the above notation; however, to be consistent with earlier chapters a binary representation is assumed when genetic operators are applied. More formally, x_i refers to the i^{th} element of the population vector, where i is a binary string in the set of all strings of length l. That is, $i \in \Omega$, where $\Omega = \{0,1\}^l$ and $\|\Omega\| = n$. In this chapter n is the number of distinct genotypes represented in the population. The symbols $\oplus$ and $\otimes$ will represent *bit-wise* e*xclusive or* and *bit-wise and* operations on binary strings, respectively. In the context of bitwise operations, $\overline{k}$ will mean bitwise complement and the notation $\|k\|$ signifies an explicit count of one bits in the bits string.

5.2.2 Modeling The CCEA with Multi-Population EGT

As stated above, the dynamical systems model I present makes the assumption that populations are infinitely large. This allows existing EA theory to be leveraged (Vose 1999) directly, as well as the more biologically-oriented EGT theory (Weibull 1992; Hofbauer and Sigmund 1998; Maynard-Smith 1982). In both cases, populations consist of an infinite number of individuals, but a finite number of n distinct possible genotypes.

With this, one can represent the state of a population at any given time by a vector in the unit simplex and changes in a population over time as dynamical system trajectories in the simplex.

This thesis investigates a class of multi-population CCEAs that are relatively straightforward applied to static optimization problems. Modeling these CCEAs on such problems affords several analytical conveniences. In addition to these, there are one or two simplifications made to ease the burden of analysis. A short discussion of some of these conveniences and simplification follows.

Symmetry

Consider a two-population cooperative coevolutionary algorithm. A common way of expressing the payoff rewards from individual interactions in the system is by using a pair of *payoff matrices*. In general, when individuals from the first population interact with individuals from the second, one payoff matrix is used (e.g. A), while the second population receives rewards defined by a second payoff matrix (e.g. B). Figure 5.2.2 below illustrates this idea. However, my model is somewhat more specifics.

$$\begin{array}{c|ccc} \textbf{A} & & \mathrm{pop}_2 & \\ \hline & 1 & 3 & 2 \\ \mathrm{pop}_1 & 2 & 5 & 3 \\ & 1 & 2 & 4 \end{array} \qquad \begin{array}{c|ccc} \textbf{B} & & \mathrm{pop}_1 & \\ \hline & 1 & 2 & 1 \\ \mathrm{pop}_2 & 3 & 5 & 2 \\ & 2 & 3 & 4 \end{array}$$

Figure 5.1: Example of payoff matrices for the rewards population 1 receives when playing population 2 (A), as well as the rewards population 2 receives when playing population 1 (B).

The approach to static optimization used here is one in which each population is assigned a specific argument of the function to represent or a partition of some total string (as in the case of binary representation), and individuals in a given population must collaborate with individuals from other populations in order to form a complete input with which to obtain a fitness value, the value of the objective function (Potter and De Jong 1994). The reader is referred back to the diagrams shown in Figures 3.3 and 3.4 on pages 31 and 31 in Chapter 3.

Recall the comments about symmetry made at the end of Chapter 3. Notice that the two payoff matrices for static single-objective optimization problems have a certain kind of symmetry—not symmetry with respect to each matrix independently (they are not necessarily symmetric), but symmetry with respect to the game itself. That is, when a specific strategy from one population is pitted against a specific strategy from the other population, the reward is the same regardless of which population one is scoring. The game is *symmetric*, $B = A^T$.

A simple two argument function serves as an example. Suppose we would like to optimize the function $f(x,y) = x^2 + y^2 + x$ using cooperative coevolution. Potential x argument values are represented in one population, and potential y argument values in a second population. In the CCEA, of course, one evolves the two populations separately (i.e., they do not interbreed), but when it comes time to evaluate an individual in the x population, as always, we will need to select collaborating individuals from the y population in order to obtain a value from the objective function. The same process is true in reverse for the y population, with respect to collaborations from x. Assuming the number of distinct genotypes for each population is finite, one can elicit a payoff matrix for the first population by simply determining the objective function values at each combination of genotypes with the collaborating population's genotypes. Since the game is symmetric, the second population uses the transpose of this matrix.

Indeed, the CCEAs discussed here are modeled well by what are known as multi-population symmetric (MPS) games (Wiegand, Liles, and De Jong 2002a). The defining characteristic of MPS games is that they are symmetric with respect to their payoff matrices. This symmetry assumption allows for some subtle simplifications of the mathematics involved. For example, simple algebraic expansion will show that the weighted average payoff of the first population is the same as that of the second, $\vec{x} \cdot A\vec{y} = \vec{y} \cdot A^T \vec{x}$. In the game-theoretic framework, the genotypes from the x population determine which row of the payoff matrix A will be used, and genotypes from the y population to determine which column of A will be used.

Another advantage to this approach is that it provides a certain clarity with respect to the algorithm's cooperative nature. When rewards are symmetric between two distinct populations, there can be no doubt that the problem *is* cooperative by nearly any definition. Individuals from the two populations *can only* succeed or fail together, due specifically to the symmetry. While it may not be true that cooperative coevolutionary algorithms *must* have a symmetric payoff, it is almost certainly true that any multi-population coevolutionary algorithm with symmetric rewards between the populations *is* cooperative. This allows me to avoid the debate discussed in Chapter 3 entirely: these algorithms are definitely cooperative ones.

Model Assumptions & Simplifications

With an EGT model comes a variety of simplifying assumptions, several of which have already been touched upon. For example, it has already been stated that the model assumes that populations are infinite in size. This assumption, while certainly important and restrictive, is not as naive as it might first appear. First of all, the model can be expanded in the future to consider populations of finite size by modeling populations as *samples* of the infinite population. Existing Markov model theory can then be applied to these augmented systems (Vose 1999; Schmitt 2003; Liekens, Eikelder, and Hilbers 2003). Second, there has been some study suggesting that finite population models with very large population sizes approximate behaviors of infinite population EC models (Vose 1999), albeit the meaning of the word "large" has admittedly only been investigated in certain constraining circumstances (Rabani, Rabinovich, and Sinclair 1998; Schmitt 2001).

Another simplification mentioned above is that the model presented here specifically describes dynamics for only two populations. While this again may seem an extreme simplification, it should be noted that the original intent of this work was to lay a foundation of study, and that this foundation would, by definition, begin with simpler systems for which behavioral understanding is more tractable. Multi-population models become complicated quickly, both in terms of their demands on our intuition, as well as the mathematics involved to model them. Modeling CCEAs with two populations is a reasonable way to gain a better fundamental understanding of the behavior of these algorithms.

In addition to these two issues, assumptions must be made as to how individuals interact in order to obtain an assessment of fitness. Although many kinds of interactions are possible, I will retain the standard EGT assumption of *complete mixing*. This means that during the evaluation phase of the algorithm, individuals in one population are assumed to have interacted with all members of the alternate population in pairwise collaborations, and vice-versa. To be clear, here the word "mixing" refers to how individuals interact, not variation. However, unless it is otherwise stated, the reader should assume that references to the word "mixing" refers to variation. With complete mixing, one not only evaluates a given member of x with every member of y for collaboration purposes, but also takes the average of the resulting fitness values. Other mixing methods are often used in practice (see Chapter 4 of this dissertation, for example) due to the computational difficulties that arise from this method; however, the theoretical model is complicated by such methods because they can introduce discontinuities and other complexities. Analysis that relaxes this assumption is the topic of future research; however, several alternative mixing models are presented at the end of this chapter as a first step toward such an analysis.

Parents in the CCEA modeled here are chosen using proportionate selection. In this case the algorithm is generational[1]. It should be noted that the EGT framework allows for analysis of coevolution using other selection methods (Ficici and Pollack 2000b).

One final important difference between the algorithm described so far and the model about to be presented should be discussed. Recall that the abstract CCEA presented in Chapter 3 on page 29 is *sequential* in the sense

[1]In a generational EA a new population entirely replaces the old population in a given step, while in a steady state EA each step replaces only one parent with one child.

that each population is processed and updated in a particular order. The EGT model that follows performs such an update simultaneously, so reflects a *parallel* update mechanism. That is, generations for the first population and the second population are processed at the same time, using information from the previous round. The net effect is that changes in one population in the middle of a round cannot affect the other population in such a model. Since both are legitimate implementation choices for the CCEA, and since there are reasons to believe that many CCEA properties are general between the two (Jansen and Wiegand 2003c), this is reasonable modeling choice.

The MPS Model

I now define the MPS dynamical system specifically, initially without variational operators. A round consists of two phases for each population: fitness assessment and selection. At an abstract level, one might write the equations as a composition of these two phases for each population. The equations below show this abstraction using the notation $\mathcal{F}$ to represent a function that assigns some vector of relative fitnesses to the strategies and the notation $\mathcal{S}$ to represent a function for selection. Note that selection must consider fitness *and* the population vector itself.

$$\mathcal{G}_x = \mathcal{S}\left(\mathcal{F}_x, \vec{x}\right) \tag{5.1}$$
$$\mathcal{G}_y = \mathcal{S}\left(\mathcal{F}_y, \vec{y}\right). \tag{5.2}$$

Biologists and economists have a more specific form of this abstraction, defined by what they call *replicator dynamics* (Weibull 1992; Hofbauer and Sigmund 1998). I make use of this model, in discrete time form, in order to describe the CCEA in the following way. Given two populations, $(\vec{x}, \vec{y}) \in \Delta^n \times \Delta^m$, representing ratios of genotypes in two infinite populations, the following equations are used to define the dynamical system for a cooperative coevolutionary algorithm (without variation) operating on these populations:

$$\vec{u} = A\vec{y} \tag{5.3}$$
$$\vec{w} = A^T\vec{x} \tag{5.4}$$
$$x_i' = \left(\frac{u_i}{\vec{x} \cdot A\vec{y}}\right) x_i \tag{5.5}$$
$$y_i' = \left(\frac{w_i}{\vec{x} \cdot A\vec{y}}\right) y_i \tag{5.6}$$

where $\vec{x}'$ and $\vec{y}'$ represent the new population distributions for the next generation. A and A^T describe the payoffs associated with each pair of possible interactions. This more specific model relates to the abstract one in a straightforward manner. Given the payoff matrix A, $\vec{u} = \mathcal{F}_x(\vec{x}, \vec{y})$ and $\vec{w} = \mathcal{F}_y(\vec{x}, \vec{y})$

For completeness, it should be noted that frequently an additive constant appears in the first equation, $\vec{u} = A\vec{y} + \omega_o$, where $\omega_o = 1 - \min(A)$, as well as a similar constant for the second equation. These constants are used to make sure elements of $\vec{u}$ and $\vec{w}$ are non-negative so that under proportionate selection the system remains invariant in the simplex. That points remain in the unit simplex as trajectories are advanced through the replicator dynamics. To keep the model simple, the payoff matrices will contain only positive values. As a result, the constants are not really necessary and will not be used.

Though variation is omitted here, it will be discussed later in the chapter.

5.2.3 Term, Tools and Problems

Before conducting the analysis, some further groundwork must be established. I begin by providing more explicit definitions for terms that are important for the discussions relating the dynamical systems ideas to the algorithm being modeled. After this, I present two specific tools that I will use to help augment formal analysis: rain-gauge validation and trajectory visualization. Finally, I describe some useful problem domains.

Terms and Concepts

One of the benefits of having a more rigorous description of the CCEA is that it provides the opportunity to be more specific in describing intuitive ideas. Let's begin by making it clear what we mean by several common dynamical systems terms.

Definition 23. *Given a function $f : S \to S$, a point $s \in S$ is said to be a* fixed point *if $f(s) = s$.*

Definition 24. *The* Euclidean length *of a vector $\vec{x} = \langle x_1, \ldots, x_n \rangle \in \mathbb{R}^n$ is $\|\vec{x}\| = \sqrt{x_1^2 + \cdots + x_n^2}$. Let $\bar{v} = \langle v_1, \ldots, v_n \rangle \in IR^n$, and let ε be a positive number. The* ε-neighborhood *$N_\varepsilon(\bar{v})$ is the set $\{\vec{x} \in IR^n : \|\vec{x} - \bar{v}\| < \varepsilon\}$, the set of points within Euclidean distance of ε of $\bar{v}$.*

Definition 25. *Let f be a map on $\mathbb{R}^n$ and let $\bar{v} \in IR^n$ be a fixed point. If there is an $\varepsilon > 0$ such that for all $\vec{x}$ in the ε-neighborhood $N_\varepsilon(\bar{v})$, $\lim_{k\to\infty} f^k(\vec{x}) = \bar{v}$, then $\bar{v}$ is a* stable fixed point.

Definition 26. *Let there be a function $f : S \to S$ and a fixed point $s \in S$, $f(s) = s$. We define the set $B(s) \subset S$ to be the set of initial points in S that eventually map to the s fixed point. The set defined by $B(s)$ is considered the* basin of attraction *of s.*

In conjunction with the dynamical systems terms, it is necessary to clarify some basic game-theoretic terms, as well. For example, the term *pure strategy* refers to a population vector that is at a basis vector ($\vec{x} = \vec{e}_i$, where $e_i = 1$ and $\forall j \neq i, e_j = 0$). For clarity I will distinguish non-pure strategy population vectors by the term *polymorphic*. In order to keep the notation clean and simple, the term $\bar{v}$ is used to refer to a fixed point $(\vec{x}, \vec{y}) \in \Delta^n \times \Delta^m$, whether polymorphic or not.

Perhaps the most important game-theoretic term that must be defined is that of Nash equilibrium (Nash 1950).

Definition 27. *A **pure Nash equilibrium** of a two player, strategic game is a pure strategy $s^*_{i,j}$ such that for every player $i \in I$ we have:*

$$\pi_i\left(s_{i,j}, s^*_{\neg i,j}\right) \leq \pi_i\left(s^*_{i,j}, s^*_{\neg i,j}\right), \ \forall s_{i,j} \in S_i$$

*A Nash equilibrium is considered a **strict Nash equilibrium** if it is unique, that is the inequality is strictly $>$.*

Less formally, a Nash equilibrium is a point such that, if either population adopts the strategy associated with the equilibrium, the interacting population cannot get a better payoff than to also play at the Nash point. As a result, the best course of action for both populations is to play at a Nash equilibrium (there may be more than one). The basic idea of this definition is the same for polymorphic Nash equilibria.

All of this terminology from the mathematical tools is useful, but there is still the question of how to relate the mathematics to the study of the algorithm in ways that address the fundamental question. For instance, what does it mean for such a system to "converge", or to "nearly converge"? In the previous chapter, the run time analysis viewpoint seems to suggest that convergence occurs with first hitting time: the algorithm has converged when it has first reached the global optimum. While this is a useful viewpoint for that kind of analysis, it is not consistent with the way most practitioners use the term. Since EAs have no guarantee of finding the global optimum in some fixed budget of time, real-world applications seldom consider the algorithm in terms of its having run until the global optimum is reached (even if it were known). Generally practitioners refer to the term loosely to mean the event that occurs when there is very little diversity left in the system, and objective progress seems to have stalled. In this dynamical systems context the term *convergence* can be defined as follows.

Definition 28. *A population vector, $\vec{x}$, in a CCEA has* converged, *or* converged to homogeneity, *when it is represented by some basis vector. $\vec{x}$ has* nearly converged *when there is some very small, positive constant ε such that $\|x - e_i\| < \varepsilon$.*

An algorithm has converged *or* nearly converged *when all its populations have done so.*

When populations $\vec{x}$ and $\vec{y}$ have both converged, I say that the those populations are *associated* or *correspond with* a payoff value $a_{i,j} \in A$, when $\vec{x} \cdot A\vec{y} = a_{i,j}$. When a CCEA has converged to a basis vector associated with the maximum value in the payoff matrix, I say that it has *converged to the global optimum*. The example below shows a sample payoff matrix with a maximum value of 5. The pure strategy population vectors $\vec{y} = \vec{x} = \langle 0, 1, 0 \rangle$ correspond with this global maximum.

$$\begin{array}{c|ccc} \mathbf{A} & & \text{pop}_2 & \\ \hline & 1 & 3 & 2 \\ \text{pop}_1 & 2 & \mathbf{5} & 3 \\ & 1 & 2 & 4 \end{array}$$

Figure 5.2: This figure demonstrates an example payoff function with a maximum value of 5. The pure strategy population vectors $\vec{y} = \vec{x} = \langle 0, 1, 0 \rangle$ correspond with this global maximum.

Modeling Iteration: Validation and Visualization

In the previous chapter all analysis was conducted directly on the actual algorithm. In an effort to provide a more general theory for understanding CCEA behavior, this chapter instead analyzes a *model* of the algorithm. Specifically, it analyzes this multi-population symmetric EGT model we've discussing. It isn't always clear how closely the model matches the real algorithm, or perhaps more foundational what the mathematical results *mean* in terms of the real algorithms.

To help bridge this gap, it is often useful to take the mathematical model and *iterate* it. By this I mean pick a random initial condition and apply the model repeatedly until I believe some limit behavior has been reached. How this end point is determined is a relatively complicated question, which I will forgo for the moment. The result of this iteration is a *trajectory* through the Cartesian product simplex space, as well as the fixed point resting place of the trajectory. With these pieces of information, projections of the trajectories can be visualized and the basins of attraction can be measured. These two tools will help us understand some of the connections between the model and the real algorithm at a less formal level.

I have defined the term basin of attraction in the usual way: the basin of attraction of a given fixed point, or any limiting behavior, is the set of initial points that will eventually map to that point, or appropriate limit behavior. Measuring the sizes of the basins of attraction of all the various limiting behaviors of a dynamical system is difficult. First of all, there is no guarantee that there is any general analytical way to do so. Second, it is typically difficult to definitively *know* all the possible limiting behaviors, much less measure their basins. Moreover, the dimensionality of the spaces of the systems in which I am interested are very large, so even when restricting attention to just the fixed points, assuming there are no cyclical or chaotic orbits, the space of potential attractors may be quite large. Fortunately, as we will shortly see, the results of my analysis will allow me to make this more tractable. This will be discussed in more detail after the analysis.

To estimate the size of a basin of attraction the simple method called a *rain gauge measure* (Alligood, Sauer, and Yorke 1996) is used. An initial point is selected uniformly at random from the product simplexes using Zuev's method (Zuev and Kahan 2001), a trajectory through the space is computed using the initial point, the system is iterated some large number of times, then the limiting behavior is examined. In my case all trajectories move to a finite set of known potential fixed points, so I maintain a histogram corresponding to these points. If the trajectory seems to have converged "very close" to a particular basis vector, I increment its value in the histogram. I then repeat this process some large number of times (5000).

While the condition "very close" is somewhat qualitative, and in *general* may not be sufficient (e.g., unstable points will push points that are "very close" away), we *can* be more comfortable with this choice if an observed trajectory approaches a known stable fixed point (which will be shown in this case). Validation studies of this nature were run for 2000 iterates, or until they were "very close" meaning within a delta of 10^{-4} in terms of variational distance. All iterates in this work met this latter criterion.

Iterating the model not only offers the opportunity to assess the relative sizes of the basins of attraction, an additional benefit is its ability to help us visualize and characterize the trajectories themselves. This can be done by plotting something similar to the so-called *takeover curves* used for standard GA analysis (Goldberg and Deb 1990). However, in this case the curves must be two dimensional in order to capture both populations. The plots are constructed by first identifying which genotypes correspond with the maximum payoff value in each population and plotting over time the proportions of these genotypes in their respective $\vec{x}$ and $\vec{y}$ population vectors. This can be seen as a top-down view of these two-dimensional takeover curves.

With these two tools, one can gain intuition about a number of things. First of all, with the visualization we can gain some limited understanding of what the population itself is doing during the search: where it is going and how various algorithmic and problem characteristics are affecting this trajectory. Second of all, with the validation we can gain some perspective on how likely trajectories are to move to particular fixed points. This helps connect us to the real algorithm by addressing the question of likelihood of global convergence.

Problem Domain

In order to conduct an analysis, I will need to do so in the context of problems or, at the very least, properties of problems. Thus, it is necessary to explain how a simple encoding of a static optimization problem can be mapped into this EGT model context. The process is quite simple.

Any two argument function can be mapped into a payoff matrix for a two population symmetric game (Wiegand, Liles, and De Jong 2002a). Doing so only requires an assumption that the fitness space is discrete. A matrix can be produced by simply enumerating all genotypic values for one population along the rows, and all genotypic values for the second population along the rows, and evaluating the given function at each ordered pair. In practice this process would be absurd (since enumerating the space would solve the optimization problem), but here the idea is an abstraction for the purpose of theoretical research.

Let us start with a straightforward problem for which it is easy to gain some intuition, and for which visualization is tractable, a simple 2D parabolic function, SIMPLEQUADRATIC.

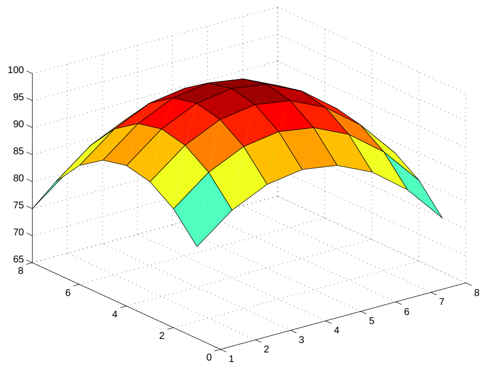
Definition 29. *Given a constant offset k and a location for the center peak (x_c, y_c), the function SIMPLEQUADRATIC $: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{R}$ is defined by*

$$\textsc{SimpleQuadratic}(x,y) = k - \left[(x_c - x)^2 + (y_c - y)^2\right].$$

In my case, the domain values for both arguments are in the interval $[1,8]$. To be specific, let us say that the maximum value, $k = 100$, is found near the center of the bowl—located at $(x_c, y_c) = (4,4)$. In all cases each argument has eight distinct genotypes represented by both populations, making a 8×8 payoff matrix (when variation is considered, this can be seen as being represented by 3-bit strings each). The figure below illustrates a surface plot of this function, as well as the payoff matrix elicited by this objective function.

This landscape is helpful precisely because of its simplicity. While it is simple, there are still interesting observable phenomena when the CCEA is applied to it, as will be clear when we look at the effects variation has on CCEA dynamics. Further, in these cases the unimodality of the simple 2D parabolic function allows more complicated dynamical behaviors (such as convergence to suboptimal, local peaks) to be ruled out as a means of explaining some of these effects.

A somewhat more interesting object of study is the MAXOFTWOQUADRATICS landscape, which presents a bimodal problem. This problem retains much of the simplicity of the SIMPLEQUADRATIC in terms of analysis, while adding the additional complexity of having two peaks. This provides the possibility of converging to a suboptimal peak in the landscape. I define a broad class of configurable landscapes below.



A	pop$_1$							
	82	87	90	91	90	87	82	75
	87	92	95	96	95	92	87	80
	90	95	98	99	98	95	90	83
pop$_2$	91	96	99	**100**	99	96	91	84
	90	95	98	99	98	95	90	83
	87	92	95	96	95	92	87	80
	82	87	90	91	90	87	82	75
	75	80	83	84	83	80	75	68

Figure 5.3: This figure depicts the SIMPLEQUADRATIC function pictorially (left), as well as the payoff matrix it elicits (right). The global maximum appears in bold in the payoff matrix.

Definition 30. *Given constant values defining the two peaks: k_1 and k_2 to define the peak heights, s_1 and s_2 to define the peak width, as well as $(\bar{x}_1, \bar{y}_1)$ and $(\bar{x}_2, \bar{y}_2)$ to define the locations of the peaks, the function* MAXOFTWOQUADRATICS $: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{R}$ *is defined by*

$$\begin{aligned} quad_1(x,y) &= k_1 - s_1 \cdot \left[(\bar{x}_1 - x)^2 + (\bar{y}_1 - y)^2 \right] \\ quad_2(x,y) &= k_2 - s_2 \cdot \left[(\bar{x}_2 - x)^2 + (\bar{y}_2 - y)^2 \right] \\ \text{MAXOFTWOQUADRATICS}(x,y) &= \max\left(quad_1, quad_2\right) \end{aligned}$$

The useful thing about this function class is that the problem can be tuned in such a way that the global peak covers much of the domain, covers an equitable portion of the domain, or covers very little of the domain relative to the local peak. This allows one to define specific landscapes where the globally optimal and locally sub-optimal peaks have different attracting potential. The parameter settings for MAXOFTWOQUADRATICS used for my investigations are defined in Table 5.2.3 on page 74.

The value for s_1 will be varied in order to control relative amount of coverage the two peaks have over the domain. The two figures on page 74 show two different instantiations of this problem class, at the extremes of s_1. The first provides one in which the global and local peaks cover more or less comparable areas of the domain, while the second shows a landscape with a broad suboptimal peak and a narrow global peak.

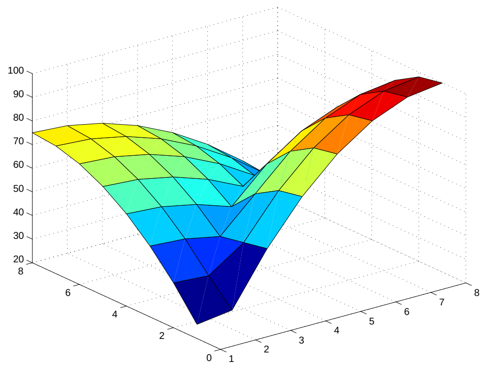
5.3 Dynamical Systems Analysis

With this model, and these definitions in place, the fundamental question can now be stated much more specifically: “Are CCEAs generally predisposed to converge to the global optimum?” Rephrased, this can be stated as follows. Starting in an arbitrary spot in the Cartesian product simplex, are trajectories likely to converge to homogeneity at, or near, the basis vector associated with the maximum payoff value? For dynamical systems models, this raises such questions as what points in the space are fixed points, what is the stability of those fixed points, and what is the relative size of basins of attraction of such points?

The remainder of this chapter will focus on attempting to answer questions about these characteristics with the tools I mentioned above, both without and with variational operators. I begin with the model as it as already been presented, without variation.

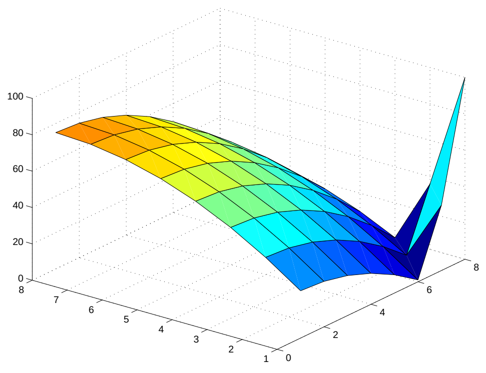
Table 5.1: This table describes the parameter settings used for the MAXOFTWOQUADRATICS function.

Parameter	**Value(s)**
k_1	100
k_2	75
s_1	$\{2,4,8,16,32,64\}$
s_2	1
$(\overline{x_1},\overline{y_1})$	(8,1)
$(\overline{x_2},\overline{y_2})$	(1,8)



A	pop$_1$							
	26	28	50	68	82	92	98	**100**
	39	38	48	66	80	90	96	98
	50	49	46	60	74	84	90	92
pop$_2$	59	58	55	50	64	74	80	82
	66	65	62	57	50	60	66	68
	71	70	67	62	55	46	48	50
	74	73	70	65	58	49	38	28
	75	74	71	66	59	50	39	26

Figure 5.4: This figure depicts the MAXOFTWOQUADRATICS function with $s_1 = 2$ pictorially (left), as well as the elicited payoff matrix (right). The global maximum appears in bold in the payoff matrix.



A	pop$_1$							
	26	25	22	17	10	1	36	**100**
	39	38	35	30	23	14	3	36
	50	49	46	41	34	25	14	1
pop$_2$	59	58	55	50	43	34	23	10
	66	65	62	57	50	41	30	17
	71	70	67	62	55	46	35	22
	74	73	70	65	58	49	38	25
	75	74	71	66	59	50	39	26

Figure 5.5: This figure depicts the MAXOFTWOQUADRATICS function with $s_1 = 64$ pictorially (left), as well as the elicited payoff matrix (right). The global maximum appears in bold in the payoff matrix.

5.3.1 Analysis without Variation

The goal of this section is to communicate a better understanding CCEA dynamics by analyzing the corresponding MPS models from a dynamical systems perspective. In particular, the focus is on population trajectories, the existence and location of fixed points and their basins of attraction. I summarize what is known for MPS models without variation, set the stage for studying the effects due to variation, and address questions about where fixed points are, and when they are stable.

Stability of Fixed Points

If variation is excluded, there is much that can be said analytically about MPS models. For example, any *strict* Nash equilibria (see Definition 27 above) must contain only pure strategies; that is, they must be at the basis vectors, the corners of the simplexes (Hofbauer and Sigmund 1998). This means that in the absence of variational operators, one can expect the populations in these systems to converge to homogeneity. It is also known, however, that mixed strategy equilibria are possible on the $\mathrm{bnd}(\Delta^n \times \Delta^m)$ when the Nash points are not strict. This can happen when there are plateaus or ridges in the objective landscape, for instance.

Indeed, there's much that can be understood about the limiting behavior of these algorithms from this model. Below, I present proofs of some useful properties about discrete time MPS systems as models of CCEAs applied to static optimization. These proofs are very instructive. From them several things are shown about when basis vector fixed points are purely stable and purely unstable. In addition, a basis is provided for understanding how certain kinds of local convergence problems can occur in a CCEA, even with infinite populations and no variation.

However, before beginning, some context and plan for the proofs should be established. The intent is to be able to make some general statements about the *Jacobian* of the system of equations generated by algebraic expansion of the replicator and selection equations (See Eq. 5.3–5.6 on page 69) evaluated at fixed points, $\bar{v}$ that are associated with the corners of the simplex. Since one can shuffle rows and columns of the payoff matrix and the game remains the same, without loss of generality I restrict the discussion to the case where $\bar{v}$ corresponds with the element $a_{1,1}$ in the A payoff matrix. A can be expressed as shown below. All other cases follow as a result.

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & \ddots & \cdots & a_{2,n} \\ \vdots & & & \vdots \\ a_{n,1} & \cdots & & a_{n,n} \end{bmatrix} \tag{5.7}$$

I start by showing that there are particular patterns that will necessarily be found in the *Jacobian* when evaluated at the fixed point, $J_v(\bar{v}_{a_{1,1}})$. This structure allows me to make assertions about the eigenvalues of the *Jacobian* in the general case. These steps lead directly to a theorem that explains that the fixed point associated with the maximum value in the payoff matrix is always stable. Several corollaries follow directly from this theorem that give more information about the nature of stability in basis vector fixed points.

Lemma 1. *Let J_v be the* Jacobian *of the system of equations generated by algebraic expansion of the two-population, n-strategy evolutionary game, where A is the payoff matrix and the replicator equations are given in Equations 5.3 – 5.6. Given the fixed point $\bar{v}_{a_{1,1}}$ associated with the $a_{1,1}$ item in A, the eigenvalues of the* Jacobian *evaluated at the $\bar{v}_{a_{1,1}}$ fixed point, $J_v(\bar{v}_{a_{1,1}})$, are the diagonal elements.*

Proof. The proof begins by partitioning $J_v(\overline{v}_{a_{1,1}})$ into four partitions. These partitions are used to show that $J_v(\overline{v}_{a_{1,1}})$ must be in a particular form, then conclude it by proving that lemma 1 follows as a result of this form.

$J_v(\overline{v}_{a_{1,1}})$ can be partitioned into four equal sized partitions B, C, D, and E shown in equation 5.8 below. Let partition B of the *Jacobian* correspond to all $\frac{\partial x_i'}{\partial x_j}$, C to $\frac{\partial x_i'}{\partial y_j}$, D to $\frac{\partial y_i'}{\partial x_j}$, and E to $\frac{\partial y_i'}{\partial y_j}$. Let's begin with the B partition.

$$J_v(\overline{v}_{a_{1,1}}) \quad = \quad \left[\begin{array}{c|c} B & C \\ \hline D & E \end{array}\right] \tag{5.8}$$

The n^{th} column and row from the matrix are omitted. This can be done because the x_n variable may be re-written using the other components in $\vec{x}$ (that is, $x_n = 1 - \sum_{i=1}^{n-1} x_i$), and because the new proportion for the x_n component in the next generation is fully specified by the equations without the redundant x_n'. Of course, a similar argument holds for the n^{th} column and row of all four partitions.

The algebraic form of $u_i = a_{i,1}y_1 + a_{i,2}y_2 + \cdots + a_{i,(n-1)}y_{(n-1)} + a_{i,n}\left(1 - \sum_{i=1}^{n-1} y_i\right)$ can be obtained from equations 5.3 and 5.4. In the B partition case the partials are taken with respect to one of the $\vec{x}$ variables, so the $\vec{y}$ values can be substituted from $\overline{v}_{a_{1,1}}$ since they will be considered constants in the derivatives. This provides a somewhat simpler algebraic form to use: $u_i = a_{i,1}y_1 + a_{i,n}(1-y_1) = a_{i,1}$. Most of the $\vec{x}$ values may also be substituted as constants as long as the partial derivatives can still legitimately be taken. Elements that fall in the i^{th} row or j^{th} column, as well as those in the first column and row are preserved in order to examine the partial—all other values in $\vec{x}$ are zero, so further simplification is possible. The x_i' algebraic form is shown below.

$$x_i' \quad = \quad \frac{a_{i,1}x_i}{x_1 a_{i,1} + x_i(a_{i,1} - a_{n,1}) + x_j(a_{j,1} - a_{n,1})}$$

If $i \neq j$ then one can substitute either 0 or 1 for x_i when taking the partial with respect to x_j. If $i \neq 1$, then it is true that $\frac{\partial x_i'}{\partial x_j} = 0$ since the numerator will remain a zero factor after the derivative. Therefore it can be said that all elements in the A partition of $J_v(\overline{v}_{a_{1,1}})$ are zero as long as $i \neq j$ and $i \neq 1$. The form of the partition is shown below in equation 5.9.

$$J_v(\overline{v}_{a_{1,1}}) \quad = \quad \left[\begin{array}{cccc} j_{1,1} & j_{1,2} & \cdots & j_{1,n-1} \\ 0 & j_{2,2} & 0 & \cdots \\ \vdots & 0 & \ddots & 0 \end{array}\right. \tag{5.9}$$

By symmetry the E partition has the same form.

The C and D partitions never have $i = j$, so their diagonals are always zero. There is only one case for these partitions that remains to be considered: when $i = 1$. This is dealt with by first taking the $i = 1$ case for C and returning to the replicator function to get the following algebraic expression after appropriate constant substitutions $u_i = a_{i,1} + a_{i,j}y_j - a_{i,n}y_j$. After simplification, the selector equation is then $x_i' = \frac{x_i\left(a_{i,1}+a_{i,j}y_j - a_{i,n}y_j\right)}{x_1\left(a_{i,1}+a_{i,j}y_j - a_{i,n}y_j\right)}$. The terms in the numerator and denominator cross out when $i = 1$ and becomes constant, the partials of which are zero. Thus all elements in the C partition (and D by symmetry) are zero.

Now that the structure of $J_v(\overline{v}_{a_{1,1}})$ is known, consider the eigenvalues of this matrix. Recall that to compute this, the characteristic equation must be solved such that $\det\left(J_v(\overline{v}_{a_{1,1}}) - \lambda I\right) = 0$. One can compute this by expansion of cofactors on the first column. From the above discussion it is known that the first column of the *Jacobian* are all zeros except elements $j_{1,1}$. The cofactors of $j_{1,1}$ is the product of the diagonal terms $J_v(\overline{v}_{a_{1,1}})$ excluding $j_{1,1}$. This can be seen by repeated application of the expansion of cofactors.

Thus, the determinate is simply the product of the diagonal terms of the matrix
$J_v(\overline{v}_{a_{1,1}}) - \lambda I$, and the roots of the characteristic equation will be the diagonal elements of the *Jacobian*. Therefore the eigenvalues of $J_v(\overline{v}_{a_{1,1}})$ are its diagonal elements. □

Lemma 2. *Given $J_v(\bar{v}_{a_{1,1}})$, the following properties are true.*

$$
\begin{aligned}
\frac{\partial x_i'}{\partial x_i'}(\bar{v}_{a_{1,1}}) &= \frac{a_{i,1}}{a_{1,1}} \; \forall i \neq 1 && (5.10)\\
&= \frac{a_{n,1}}{a_{1,1}} \; i = 1 && (5.11)\\
\frac{\partial y_i'}{\partial y_i'}(\bar{v}_{a_{1,1}}) &= \frac{a_{1,i}}{a_{1,1}} \; \forall i \neq 1 && (5.12)\\
&= \frac{a_{1,n}}{a_{1,1}} \; i = 1 && (5.13)\\
& && (5.14)
\end{aligned}
$$

Proof. Again, begin with partition B of $J_v(\bar{v}_{a_{1,1}})$. Returning to the replicator Equation 5.6 after appropriate constant substitution provides $u_i = a_{i,1}y_1 + a_{i,n}(1-y_1) = a_{i,1}$. There are two cases: $i \neq 1$ and $i = 1$. In the fist case, again retaining x_i for the partial, the remaining values for $\vec{x}$ are substituted as constants and obtain the following selection equation and subsequent partial derivation.

$$
\begin{aligned}
x_i' &= \frac{x_i a_{i,1}}{x_1 a_{1,1} + x_i(a_{i,1} - a_{n,1})}\\
\frac{\partial x_i'}{\partial x_i'}(\bar{v}_{a_{1,1}}) &= \frac{a_{i,1}x_1 a_{1,1}}{(x_1 a_{1,1} + x_i a_{i,1} - x_i a_{n,1})^2}\\
&= \frac{a_{i,1}}{a_{1,1}}
\end{aligned}
$$

In the case where $i = 1$, x_i is not preserved, so the following is obtained after relevant substitution.

$$
\begin{aligned}
x_i' &= \frac{x_1 a_{1,1}}{x_1 a_{1,1} + a_{n,1} - x_1 a_{n,1}}\\
\frac{\partial x_i'}{\partial x_i'}(\bar{v}_{a_{1,1}}) &= \frac{a_{1,1}a_{n,1}}{(a_{n,1} - a_{1,1} - a_{n,1})}\\
&= \frac{a_{n,1}}{a_{1,1}}
\end{aligned}
$$

The proof for the $\frac{\partial y}{\partial i} i$ case can be obtained from the D partition by symmetry. □

Theorem 5. *Let m be a unique maximum value in A, $m = \max(A)$. Given a non-polymorphic fixed point $\bar{v}$, if the payoff value $a_{i,j} = m$ and $\bar{v}$ corresponds with $a_{i,j}$, then $\bar{v}$ is a stable fixed point.*

Proof. Given $a_{i,j} = m$ and that $a_{i,j}$ can be moved into position $a_{1,1}$, then all eigenvalues are < 1 by lemma 2. □

Noting in lemma 2 that all the eigens for a particular $\bar{v}$ are ratios of A values at some fixed column or row, and applying the same juxtaposition logic from Theorem 5, several interesting corollaries can be derived. For example, it follows directly that any $a_{i,j}$ that is the unique maximum value of the i^{th} row and j^{th} column of A is also a stable fixed point. Additionally, any $a_{i,j}$ that is the minimum value of the i^{th} row and j^{th} column of A is a purely *unstable* fixed point (meaning *all* its eigenvalues are > 1). With these, I can bound the number of stable and unstable basis vector fixed points in the system in general. These corollaries are stated more formally below.

Corollary 1. *From Theorem 5 it follows directly that if*

$$
\begin{aligned}
a_{i,j} &> a_{l,j} \; \forall l \neq i, \; \textit{and}\\
a_{i,j} &> a_{i,k} \; \forall k \neq j,
\end{aligned}
$$

then $\bar{v}_{a_{i,j}}$ is a stable fixed point.

Corollary 2. *As per corollary 1, lemma 2 and theorem 5 dictate that if*

$$\begin{aligned} a_{i,j} &< a_{l,j}\ \forall l \neq i, \ \textit{and} \\ a_{i,j} &< a_{i,k}\ \forall k \neq j, \end{aligned}$$

then $\overline{v}_{a_{i,j}}$ is a purely unstable fixed point. The global minimum value, $\min(A)$ *is always purely unstable. Additionally any other fixed points at the corner of the simplex which are neither purely unstable nor stable will be an unstable saddle point.*

Corollary 3. *From corollaries 1 and 2 we also know that the maximum number of stable basis vector fixed points is n and the minimum number of stable basis vector fixed points is* 1. *The same rule is also true for the number of purely unstable fixed points. Therefore the number of unstable saddle basis vector fixed points must be at least $n^2 - 2n$.*

What does this analysis explain about convergence and optimization? As I made clear in Definition 28 above, when trajectories limit to a fixed point at the basis vector, this corresponds to the populations becoming homogeneous. The question is, where can this occur in terms of the fitness landscape? Recall that in our MPS model of cooperative coevolution, the payoff matrix is really just a quantized version of the fitness landscape. Given this, perhaps the most important thing worth noting is that a form of *local* convergence is possible even with an infinite population, no variation, and unique values in the fitness landscape. Trajectories can fall to basis vectors that correspond with suboptimal fitness values. This is not possible in the simple genetic algorithm under the very same assumptions (Reeves and Rowe 2002; Vose 1999).

However, knowing the stability of a fixed point in a system does not necessarily indicate how likely it will be reached by any arbitrary initial condition, unless more is known about the dynamical system (Hofbauer and Sigmund 1998). Fortunately, even without the deeper understanding required for formal proofs, considerable insight into these issues can be obtained using the model validation and visualization techniques discussed earlier. In the following section I summarize these insights for MPS models without variation.

Population Trajectories and Basins of Attraction

Let us now turn to the visualization and validation tools mentioned previously to help clarify the meaning of the formal results. Recall that visualization of these systems involves iterating the model until the fixed point is reached, then displaying its trajectory in terms of a projection offered by looking only at the component vectors associated with the global maximum. The visual effect corresponds to a form of 2D takeover curve and helps one understand where trajectories are going relative to the global peak. A rain-gauge measure can be taken for the relative sizes of the basins of attraction of the fixed points by doing this many times and recording where the final resting points.

Of particular interest is the size of a fixed point's basin of attraction relative to the other fixed points to which trajectories go. Since I've now some idea that these are very likely to be basis vectors (in the absence of variation), the question of global convergence can be addressed in a way that makes sense relative to the problem space itself. In other words, viewing convergence to a basis vector as a collapse to homogeneity in both populations and knowing the relative sizes of the basins of attraction of each possible pair of homogeneous populations should give us insight into how likely it is that a pair of random initial populations will converge to some particular pair of homogeneous populations.

Indeed, there is reason to believe that the size of the basins of attraction of a fixed point indicated by a basis vector has more to do with relative local column and row values in the payoff matrix than how large the specific payoff value is at that point. The idea is that the *joint distribution* of the rewards for a given strategy in one population as defined by the set of *possible* collaborating strategies from the alternative population is more attractive for trajectories than the *specific* value of the optimal collaboration for that strategy. In other words, broad and suboptimal peaks will pull trajectories away from taller, more narrow peaks. This is the form of local convergence in CCEAs mentioned above, and touched upon in other studies (Ficici and Pollack 2000c; Wiegand, Liles, and De Jong 2002a).

Recall that measuring the basins of attraction can be quite complicated in general. In this case though, there are three things we know, or can do, to make this a more tractable problem. First, some useful properties are already known about these systems that help. For instance, it is known that as long as the maximum values on the rows and columns are unique, the only strict Nash equilibria are at the basis vectors. Second, knowing this, I can construct a problem such that this property is true of my payoff matrix. Finally, I can apply the validation method already discussed in order to obtain this measure.

As an example, consider the SIMPLEQUADRATIC function previously discussed. In particular, I take the 8×8 payoff matrix elicited for this function and iterate the MPS model many times until convergence is reached. Having done so, the expected result that all trajectories lead to the basis vector fixed point associated with the global optimum was obtained. That is, when an initial starting state for the populations is chosen at random, the model predicted that a CCEA algorithm will converge to homogeneity at the global maximum of this function.

Figure 5.6 illustrates this for several example trajectories. Each curve begins at the point indicated by a hollow circle, and terminates at the point indicated by an "x" (in this case all of these terminate in the corner). Every 100 steps of the trajectory are marked on the curves to get an idea of the rate of progress of the curves, though all points on the plot represent steps produced by the model. Trajectories that converge near the global optimum appear as thicker lines than those that do not. In this way, one can track the proportions of the components associated with the maximum value over time as they move from the initial population configurations toward its ultimate limit behavior. Asymmetries in the projected trajectories are due to asymmetric differences in the sample of initial points or asymmetries that exist in the function itself (the latter of which, in this case, are minimal).

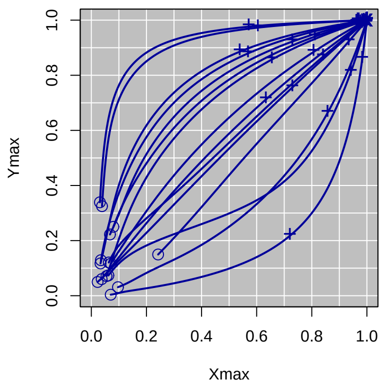


Figure 5.6: 2D Takeover plot for trajectories operating on the simple quadratic problem with no variation. Curves begin at points indicated by hollow circles, and terminates at the point indicated by an "x". Every 100 steps of the trajectories are marked.

The result of the of this is expected. But what of the fact that analysis reveals that there can be other stable basis vector fixed points than the one associated with the global maximum, and what of the problem of consensus caused by cumulative weights along joint distributions in the payoff matrix mentioned in the previous chapter? Indeed, the MAXOFTWOQUADRATICS problem is constructed so that trajectories are more attracted to the suboptimal peak than to the global maximum. This is done by making sure the rows and columns in the payoff matrix corresponding with the suboptimal local peak are significantly biased over the row and column values corresponding with the global peak. Ranging the s_1 parameter in the MAXOFTWOQUADRATICS problem presented above makes this fact palpably clear.

Consider all 6 example functions from the MAXOFTWOQUADRATICS class presented above resulting from varying $s_1 \in \{2,4,8,16,32,64\}$. Again, I use these functions to elicit 8×8 sized payoff matrices; however, varying the parameter now results in different amounts of coverage between the two peaks. Table 5.3.1 below describes the percentage of domain that the global peak covers as s_1 is increased.

Table 5.2: This table describes the percentage of domain points in MAXOFTWOQUADRATICS that are covered by the global peak as the s_1 parameter is varied.

s_1	**Coverage of global peak**
2	43.75 %
4	29.69 %
8	17.19 %
16	12.50 %
32	6.25 %
64	4.69 %

The rain gauge method described above was also run for each of these problems. The result was that in all cases trajectories converged to basis vectors associated with one of the two peaks, but not necessarily the global peak. Note that the relative difference in height between the peaks remains the same in all functions, regardless of s_1 (refer again to the figures on page 74).

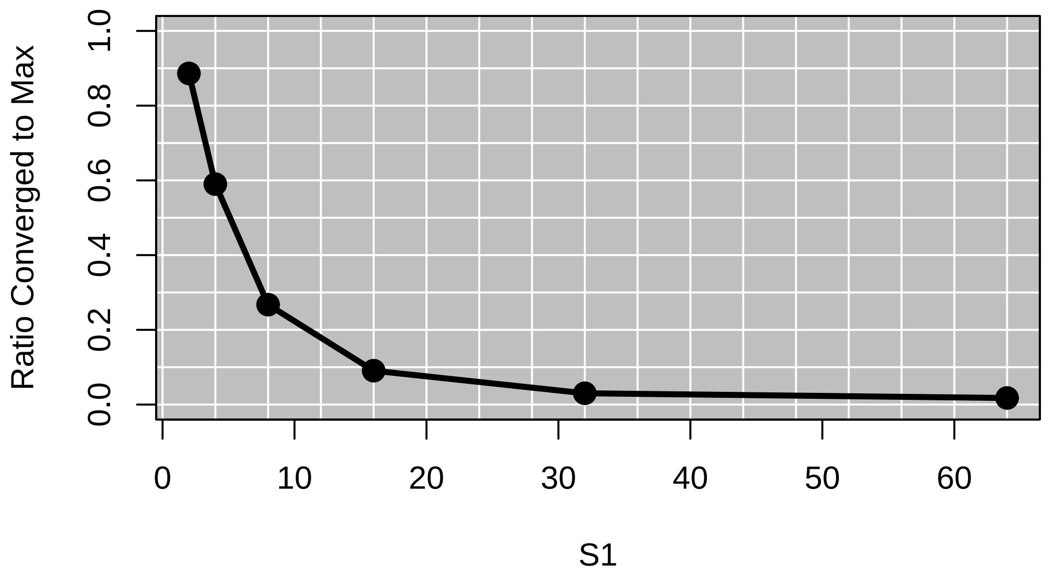


Figure 5.7: This graph shows the ratio of trajectories that converged to the basis vector associate with the global maximum of the MAXOFTWOQUADRATICS function as the s_1 parameter is varied in $\{2,4,8,16,32,64\}$.

Looking more closely at where the trajectories are actually going helps these results. Figure 5.8 on page 5.8 illustrates trajectory behavior for a subset of the those initial conditions for each of the problems generated by varying s_1. The panels appear in order from left to right, top to bottom. Some minimal amount of initial probability in the components associated with the global peak is necessary to converge to the global optimum.

As the coverage of the global peak diminishes, we begin to see the effects of the increased attraction to the local peak. This justifies our intuition that cumulative payoff values local to some suboptimal maxima can distract trajectories from finding the global peak when they are sufficiently large.

5.3.2 Analysis with Variation

In order to extend the MPS framework to model variational operators, the methods outlined by Vose (1999) have been employed. Hence, the dynamical system becomes a composition of the original model and a variational mixing function, $\mathcal{M}$. For simplicity, I assume that the variational operators are the same for both populations and the populations have the same number of distinct genotypes, so the same $\mathcal{M}$ can be used for both populations. With these assumptions the abstract formulation of the MPS model now becomes: $\mathcal{G}_x = \mathcal{M}\left(\mathcal{S}\left(\mathcal{F}_x, \vec{x}\right)\right)$ and $\mathcal{G}_y = \mathcal{M}\left(\mathcal{S}\left(\mathcal{F}_y, \vec{y}\right)\right)$.

Mixing can be done in any number of ways, so this abstract formulation does not commit to a particular representation; however, to build concrete mathematical models I will need to be more specific—and doing so will imply particular representations for individuals, as well as particular operators. Remaining consistent with earlier chapters, I assume that the representation is binary, and focus on two common operators for such representation (one for mutation and one for crossover). The $\mathcal{M}$ function is constructed for these operators.

Let's start by enumerating what the probability is that the all zero string will be produced by any arbitrary pair of individuals acting as parents. It is clear that an $n \times n$ matrix, M, results from such an enumeration. Supposing one has such a matrix, obtaining the probability that the all zero string is generated after replication and variation is a relatively straightforward exercise:

$$x''_0 = \sum_{u,v \in \Omega} x'_u x'_v M_{u,v} \tag{5.15}$$

$$y''_0 = \sum_{u,v \in \Omega} y'_u y'_v M_{u,v} \tag{5.16}$$

Variation works as follows. Parents are selected using Eq. 5.3–5.6. The resulting probabilities are used for $\vec{x}'$ and $\vec{y}'$. The $M_{u,v}$ value represents the likelihood that the all zero string is constructed or not destroyed when u and v are parents. Finally, the sum over all such combinations is taken. To produce a value for any child string, not just the all zero string, the mixing matrix is permuted (see Vose (1999) for more details). Given this, the next generation's population states (now notated $\vec{x}''$ and $\vec{y}''$) can be obtained using the following equations:

$$x''_k = \sum_{u,v \in \Omega} x'_u x'_v M_{u\oplus k, v\oplus k} \tag{5.17}$$

$$y''_k = \sum_{u,v \in \Omega} y'_u y'_j v M_{u\oplus k, v\oplus k} \tag{5.18}$$

Computing values for the mixing matrix, M is involved. The reader is referred to Vose (1999) for the derivation and motivation. Given two parents, u and v, what is the probability that the all zero string is constructed, or survives? To answer this, we first note that masks can be used to instantiate the effect of a particular crossover or mutation event. To determine probabilities for a given M_{uv}, a distribution across all such masks will be necessary. To be consistent with existing work, I use the symbol χ to represent the distribution of crossover masks and μ to represent the distribution of mutation masks. Modeling particular genetic operators involves eliciting these two distributions. The probability M_{uv} can be obtained as follows.

$$M_{uv} = \sum_{u,v,k \in \Omega} \mu_u \mu_v \frac{\chi_k + \chi_{\bar{k}}}{2} \cdot I(i,j,u,v,k) \tag{5.19}$$

$$I(i,j,u,v,k) = \begin{cases} 1 & \text{if } \left((i \oplus u) \otimes k\right) \oplus \left((j \oplus v) \otimes \overline{k}\right) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.20}$$

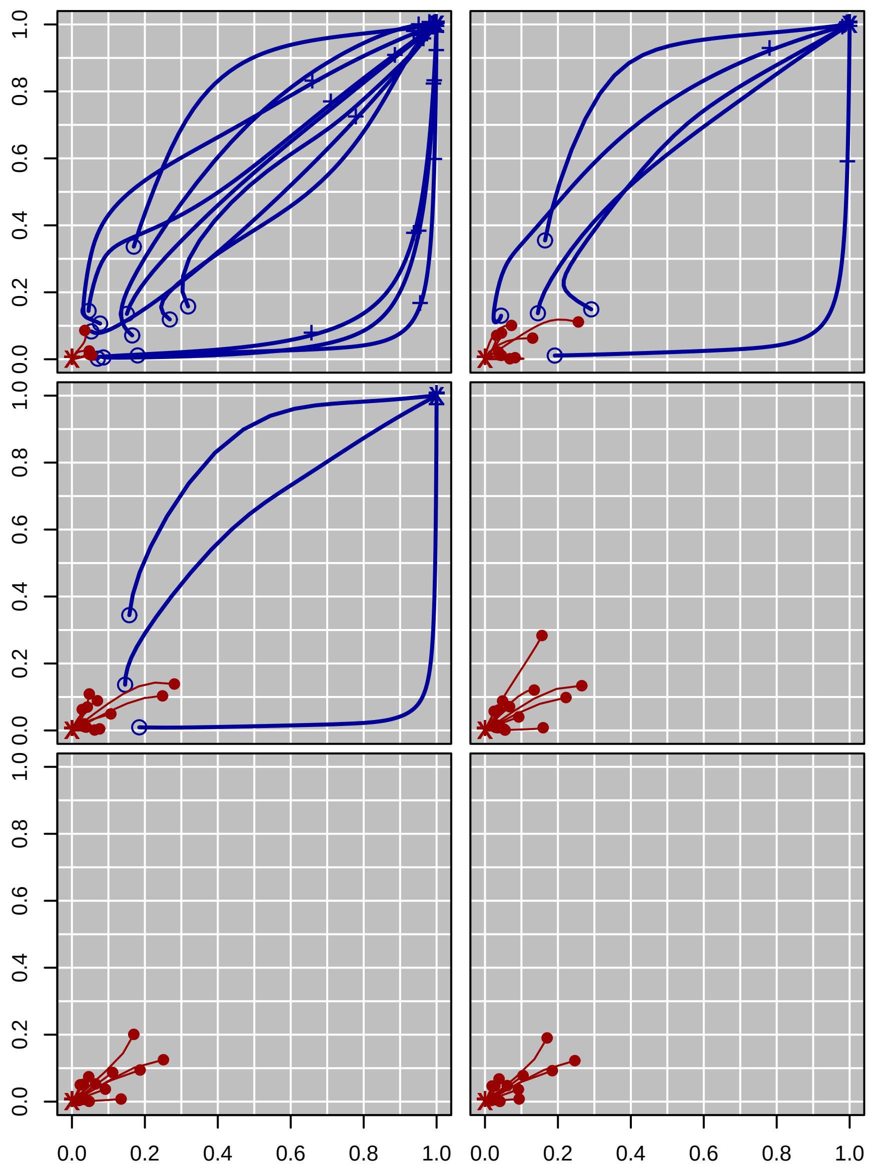


Figure 5.8: MPS trajectories on MAXOFTWOQUADRATICS with $s_1 \in \{2,4,8,16,32,64\}$. The panels appear in order, left to right and top to bottom. Open circles represent initial points for trajectories converging to global maximum. Closed circles represent initial points bound for smaller local maximum.

The meaning of the indicator function I described in Equation 5.20 is this: indicate whether or not the all zero string is formed when the i mutation event mask is applied to parent u, the j mutation event mask is applied to parent v, and the k crossover event mask is used for the recombination of those two parents. Crossover produces two children, so $\overline{k}$ is needed to represent the reflected side of the recombinatoric symmetry. The model assumes only one of the two children is used, thus the expected value is taken between the masks k and $\overline{k}$. The reader is referred to Vose (1999) for a more in depth explanation of how these mixing matrices are constructed for particular operators that are not discussed here. I will illustrate how M can be created using two specific operators, but first let's look more formally at the more general MPS model under mixing.

Fixed Points under Mixing

There is little doubt that variational operators significantly change the underlying dynamical system of a CCEA, just as they do in a traditional EA. Not only can the limiting properties of the fixed points of the system change, but the location of the fixed points themselves can change. To see this, note that for a fixed point of the MPS model to *also* be a fixed point of the variational model, it must be true that $\vec{x}'' = \vec{x}' = \vec{x}$ and $\vec{y}'' = \vec{y}' = \vec{y}$. For the purpose of this fixed point discussion, Equations 5.17 and 5.18 are re-written as follows.

$$x''_k = \sum_{u,v\in\Omega} x_u x_v M_{u\oplus k, v\oplus k} \tag{5.21}$$

$$y''_k = \sum_{u,v\in\Omega} y_u y_v M_{u\oplus k, v\oplus k} \tag{5.22}$$

As we saw earlier, the fixed points of MPS models without variation were basis vectors. What happens when variation is added? Suppose we are interested in the basis vectors associated with payoff value a_{pq}, that is $x_i = 0, \forall i \neq p, x_p = 1$ and $y_i = 0, \forall i \neq q, y_q = 1$. If such is the case, then the resulting values from the above equations are always 0, except when $u = v = p$ for the first case, and $u = v = q$ for the second case. Thus, the next point in the trajectory, when starting at the basis vector is the following:

$$x''_k = M_{p\oplus k, p\oplus k} \tag{5.23}$$

$$y''_k = M_{q\oplus k, q\oplus k} \tag{5.24}$$

There are two important things to note about this observation. First, one can ascertain the $\mathcal{G}^1_x(\vec{x},\vec{y}), \mathcal{G}^1_y(\vec{x},\vec{y})$ step of the MPS model from any arbitrary basis vector point *from just the mixing matrix*. Second, resulting values turn out to be diagonals of the p^{th} and q^{th} permutation of the mixing matrix for $\mathcal{G}_x$ and $\mathcal{G}_y$, respectively. If this diagonal is equal to the original bases, then the fixed point of the original system is also a fixed point of the system under variation, and if not then it is not. As I will discuss in the following sections, this is true for almost all traditional crossover operators and never true for traditional (non-zero) mutation.

Assessing fixed point stability under mixing is harder than without variation, since the fixed points may now be in the interior of the simplex product. This means that it becomes necessary to simultaneously solve the collective system for $\mathcal{G}_x(\vec{x},\vec{y}) = \vec{x}$ and $\mathcal{G}_y(\vec{x},\vec{y}) = \vec{y}$. However, when one is certain that the basis vectors are fixed points, even under mixing, one can evaluate them as I did in the previous paragraph.

Stability of Fixed Points

Unlike the MPS models without variation, the stability of fixed points of the model under variation is a function of M. Moreover, this dependence is not due simply to the inclusion of M itself into the model, but also from the resulting non-linearity added from the crossover operation. The proof of this is trivial, but I offer it for the sake of completeness.

Theorem 6. *Let $\bar{v}$ be a fixed point of the cooperative coevolutionary algorithm with variation as described by equations 5.17 and 5.18. The stability of such a fixed point will depend on the specific values in the mixing matrix, M.*

Proof. To assess the stability of a fixed point, one must first know what the fixed point is. While some of the fixed points are known for crossover, this is not necessarily true for mutation (or for mutation *and* crossover). Still, assuming one knew what the fixed point was, the *Jacobian* of the system could be evaluated at that fixed point. Let's look at one term $\frac{\partial x_k''}{\partial x_l}$, but first let's expand and re-write the equations 5.17 and 5.18. The variable f is used for the convenience of notational simplification only.

$$\begin{aligned}
x_k'' &= \sum_{u,v\in\Omega} \frac{(A\vec{y})_u x_u (A\vec{y})_v x_v M_{u\oplus k,v\oplus k}}{\left(\vec{x}^T A\vec{y}\right)^2} \\
&= \frac{\sum_{u,v\in\Omega}(A\vec{y})_v x_u (A\vec{y})_v x_v M_{u\oplus k,v\oplus k}}{\left(\vec{x}^T A\vec{y}\right)^2} = \frac{f}{\left(\vec{x}^T A\vec{y}\right)^2} \\
\frac{\partial x_k''}{\partial x_l} &= \frac{\frac{\partial f}{\partial x_l}\cdot\left(\vec{x}^T A\vec{y}\right)^2 - f\cdot\frac{\partial}{\partial x_l}\left(\vec{x}^T A\vec{y}\right)^2}{\left(\vec{x}^T A\vec{y}\right)^4} \\
&= \frac{\frac{\partial f}{\partial x_l}\cdot\left(\vec{x}^T A\vec{y}\right)^2 - f\cdot 2\left(\vec{x}^T A\vec{y}\right)(A\vec{y})_l}{\left(\vec{x}^T A\vec{y}\right)^4}
\end{aligned}$$

Taking the partial of f now results in the following.

$$(A\vec{y})_l \sum_{i\in\Omega,i\neq l} (A\vec{y})_i x_l M_{i\oplus k,l\oplus k} + 2(A\vec{y})_l^2 x_l M_{l\oplus k,l\oplus k}$$

By symmetry, the same is true for $\frac{\partial y_k''}{\partial y_l}$. From this we learn that both terms in the numerator depend on values from M, and that they cannot be eliminated. □

Although this was perhaps already obvious, the proof is instructive since it illustrates the fact that the non-linearities introduced by equations 5.17 and 5.18 presents a far more complicated expression that requires that change in any given component depends on all the components from *both* populations. This differs from the model without variation, since in that case taking the derivative eliminated all components of one population from the expression. Therefor, its dependence is due not just to the explicit presence of the values in M, but also due to the non-linearities created by its inclusion.

Parameterized Uniform Crossover

Let's look at the effects of crossover alone. In this case, for most types of crossover, the probability of obtaining some string k when crossing over two identical parents is 0 unless the parents are themselves k. When k is crossed-over with k, the probability that the resulting child is also k is 1. Thus the diagonal of the k^{th} permutation of M is always 1 at m_{kk} and zero elsewhere. This implies that the pairs of basis vectors forming fixed points of the MPS model without variation are still fixed points under crossover. This is consistent with an understanding of the effect of crossover on a totally homogeneous population (namely that there is no effect).

In order to study the effects of crossover on population trajectories and basins of attraction, one needs to complete the MPS model by selecting a particular crossover operator and constructing the corresponding mixing matrix M. I model parameterized uniform crossover (Syswerda 1989; Spears and De Jong 1991). Here p_c represents the probability that a crossover event will occur. If a crossover event occurs, two individuals are mated and each bit position is considered independently for potential exchange. The parameter p_s represents the probability that the values at a given bit position will be exchanged between the mates. Therefore $p_s = 0.5$ corresponds with traditional uniform crossover.

Recall that the method described above models particular crossover events using a mask over the bit strings and a crossover operator is modeled by describing a distribution across such masks. For parameterized uniform crossover, this distribution is obtained as follows.

$$\chi_i = \begin{cases} p_c \left(p_s^{\|i\|} (1 - p_s)^{l - \|i\|} \right) & i > 0 \\ 1 - p_c + p_c (1 - p_s)^l & i = 0 \end{cases} \tag{5.25}$$

Having created an MPS model that includes parameterized uniform crossover, validation and visualization of population trajectories through the mathematical model is now possible. Let's consider again the SIMPLEQUADRATIC problem and assess the effects of crossover on population trajectories and basins of attraction. Since it is known that the basis vectors of the Cartesian product of the unit simplexes are also fixed points of this model (though not necessarily stable in the same places), rain-gauge measures can be obtained. However, one must be careful to account for any trajectories that do not converge to a basis vector, since the existence of interior fixed points have yet to be formally eliminated as a possibility as it was without variation. As it turned out, all of the trajectories converged to a basis vector of one sort or another.

Initially, p_s was set to 0.5 to obtain pure uniform crossover. The rate of crossover, p_c, was varied between 0.0 and 1.0. All trajectories moved to one of four basis vectors, those vectors associated with the $a_{4,4}$, $a_{4,5}$, $a_{5,4}$, and $a_{5,5}$ payoff values (the center four cells of the payoff matrix, from the top left as described in equation 5.7 and page 75). Example results are shown in Table 5.3. This shows the percentage of trajectories from the randomly chosen initial populations that go to each of these four basis vectors. More specifically, it shows that as the rate of crossover is increased, the number of initial points that eventually converge to a fixed point associated with the global maximum shrinks. In other words, the measure of the size of the basin of attraction of that fixed point associated with the global peak is reduced by increasing crossover.

The reader should be careful not to draw too many conclusions about the significance of the "nearness" of these fixed points from a topological point of view. From the simplex product space, every basis vector is a distance of either 1.0 or $\sqrt{2}$ from every other basis vector, $\sqrt{2}$ is the largest distance possible between any two points in the space. Moreover, as long as the relationships between the strategies of the game remain the same, one could permute the payoff matrix and the underlying non-variational dynamics would not change. Thus it is possible to imagine the very same model with resulting measures that seem topologically farther apart. The relationship almost certainly has to do with the level of fitness on a given row and column relative to other strategies and the topological relationships are established by the representation assumed under the operator modeled.

Table 5.3: Rain gauge results of model validation studies on MPS cooperative coevolution model with uniform crossover. The tables represent the measure of the basins of attraction of the fixed points associated with the $a_{4,4}$, $a_{4,5}$, $a_{5,4}$, and $a_{5,5}$ payoff values. The probability of crossover, p_c is varied in $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$.

$p_c = 0.0$	100.0 %	0 %	$p_c = 0.2$	84.8 %	7.7 %
	0 %	0 %		7.0 %	0.5 %
$p_c = 0.4$	79.0 %	10.1 %	$p_c = 0.6$	75.9 %	11.0 %
	10.0 %	0.9 %		11.3 %	1.8 %
$p_c = 0.8$	73.2 %	12.0 %	$p_c = 1.0$	71.6 %	13.1 %
	12.5 %	2.3 %		12.8 %	2.5 %

I also looked at parameterized uniform crossover when $p_s = 0.2$ and $p_c = 1.0$. The results for this showed that the $(4,4)$ basin captured roughly 62.6% of the trajectories, while the $(4,5)$, $(5,4)$, and $(5,5)$ basins captured 16.2%, 17.0% and 4.2% of the trajectories, respectively. Again, all trajectories found their way to one of these four basis vectors.

These are interesting results. They suggests that it is possible that previously unstable basis vector fixed points become stable attractors under crossover. In a sense, crossover seems to "distract" trajectories from always converging to the basis vector associated with the maximum value. To get a sense for why this might be, let's look again at the 2D takeover plots shown in Figure 5.9 on page 87. In this case, $p_c \in \{0.00, 0.05, 0.15, 0.15, 0.20, 0.25\}$.

There seems to be some kind of stretching transformation on the outer corners of the product simplex going on as a result of crossover. As the rate of crossover increases, the trajectories are pushed away from the center and move toward the edges at a much faster rate. In some cases these trajectories are drawn to basis vector fixed points (homogeneous populations) not associated with the global optimum. This behavior is observed in CCEA applications as well where one population converges much faster than the other and reduces the dimensionality of the search by collapsing the space in which trajectories can pass to a face of the Cartesian product simplex. This corresponds to a reduction of the space to the unit simplex for the second population, still evolving population.

This is most likely explained by the accelerating effect crossover can have on population convergence (Menon 2002; Rabani, Rabinovich, and Sinclair 1998). This acceleration is applied asymmetrically, since the initial conditions of the populations are almost certainly asymmetric. To test this hypothesis, I ran an additional experiment in which all initial points for $\vec{x}$ were chosen uniformly at random from the unit simplex, and all initial points for $\vec{y}$ were set symmetrically, $\forall i\,(\vec{y})_i = (\vec{x})_i$, where $(\vec{x})_i$ is the i^{th} initial point. Although not all trajectories converge to proper basis vector (the corner associated with the maximum value), *none* converged sooner in one population than in the other. Figure 5.10 illustrates these results for different values of p_c.

Bit-flip Mutation

In a similar fashion, the mixing matrix M for the bit-flip mutation operator can be constructed. In this case, the diagonal elements of M cannot be basis vectors as long as $p_m \neq 0$ and $p_m \neq 1$ since a population that is completely converged cannot remain so after mutation (in an infinite population model). Therefore it is fair to conclude that the basis vectors are no longer fixed points under mutation.

Modeling bit-flip mutation is much simpler than modeling crossover, though analysis is much more difficult. Again a mask is used to apply a specific mutation event and the distribution of mutation masks models the operator as a whole. With bit-flip mutation this is determined using a simple binomial distribution

$$\mu_i \quad = \quad p_m^{\|i\|} \left(1 - p_m\right)^{l - \|i\|} \tag{5.26}$$

where p_m represents the independent probability that a bit in a given position will be flipped. A careful review of the mixing formula will confirm that as long as $p_m \neq 0$ and $p_m \neq 1$, the diagonal of any valid permutation of the mixing matrix cannot be a basis vector. This makes sense: a population that is completely converged cannot remain so after mutation.

Unfortunately, this means that rain gauge measures of the type I used so far are not practical. Instead, I can measure the distance of the point to the basis vector associated with the maximum payoff value. I ran trajectory studies with p_m set to a variety of values. The result was that all the trajectories converged very close to the same fixed point in all cases for a given p_m value, all 5000 initial conditions mapped to the same interior point within an error radius of 10^{-4}. Nevertheless, this point moved into the interior of the simplex product as mutation was increased. Figure 5.11 shows the 2D takeover plots for a few of these runs. The reader should note that the distortions of the trajectories due to mutation are different than those produced by crossover.

This behavior matches intuition, as well as known results for the simple GA. As mutation is increased, the limiting behavior is characterized as a distribution of population states that spread out away from the peak into

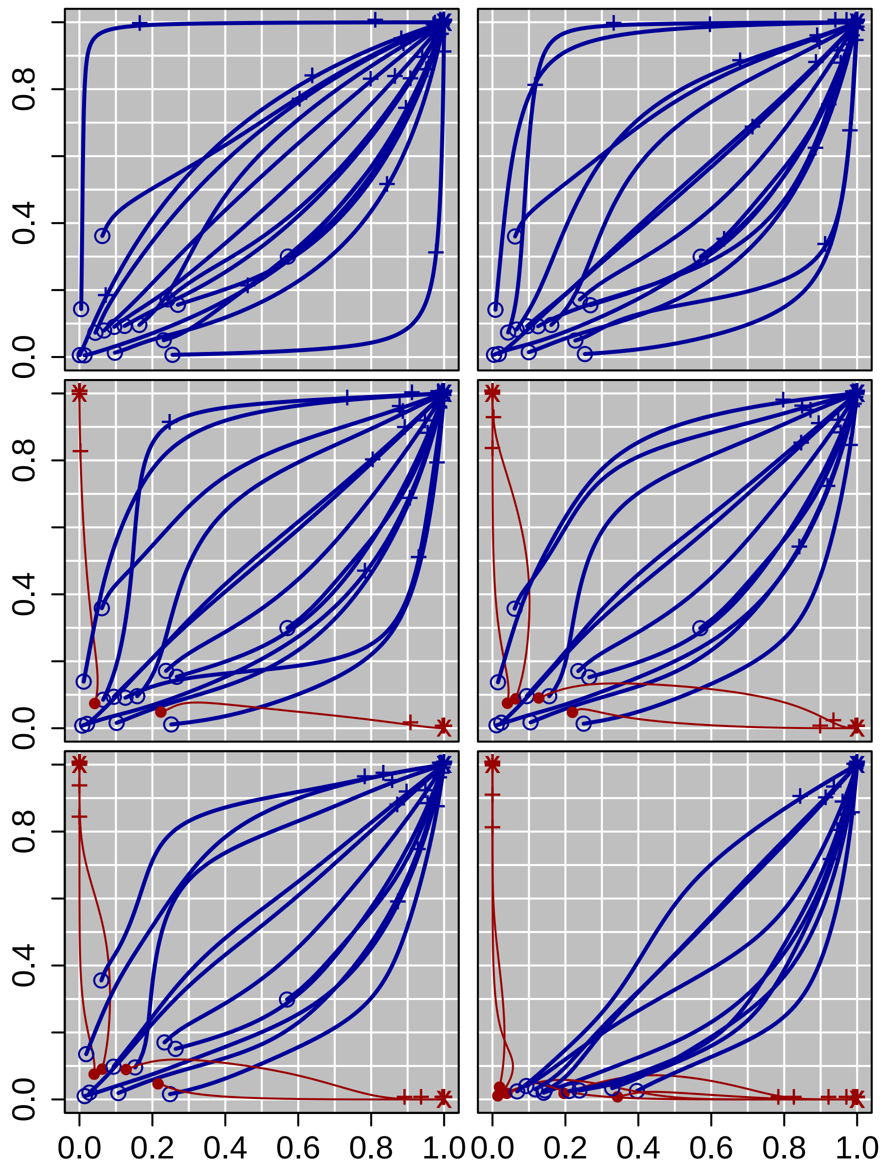


Figure 5.9: 2D takeover plots for MPS trajectories in with uniform crossover. Reading from the left to the right, top to bottom $p_c = \{0.00, 0.05, 0.10, 0.15, 0.20, 0.25\}$.

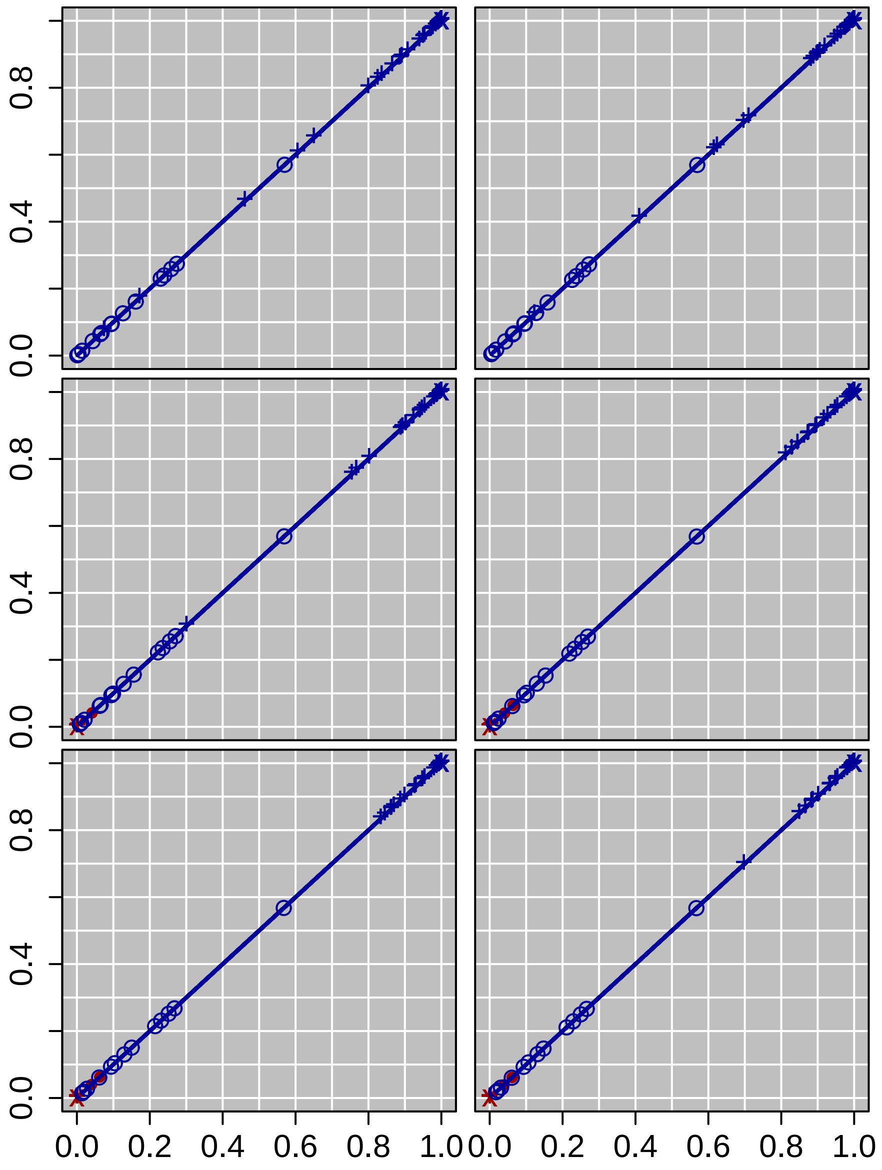


Figure 5.10: 2D takeover plots for MPS trajectories in with uniform crossover with symmetric initial conditions. Reading from the left to the right, top to bottom
$p_c = \{0.00, 0.05, 0.10, 0.15, 0.20, 0.25\}$.

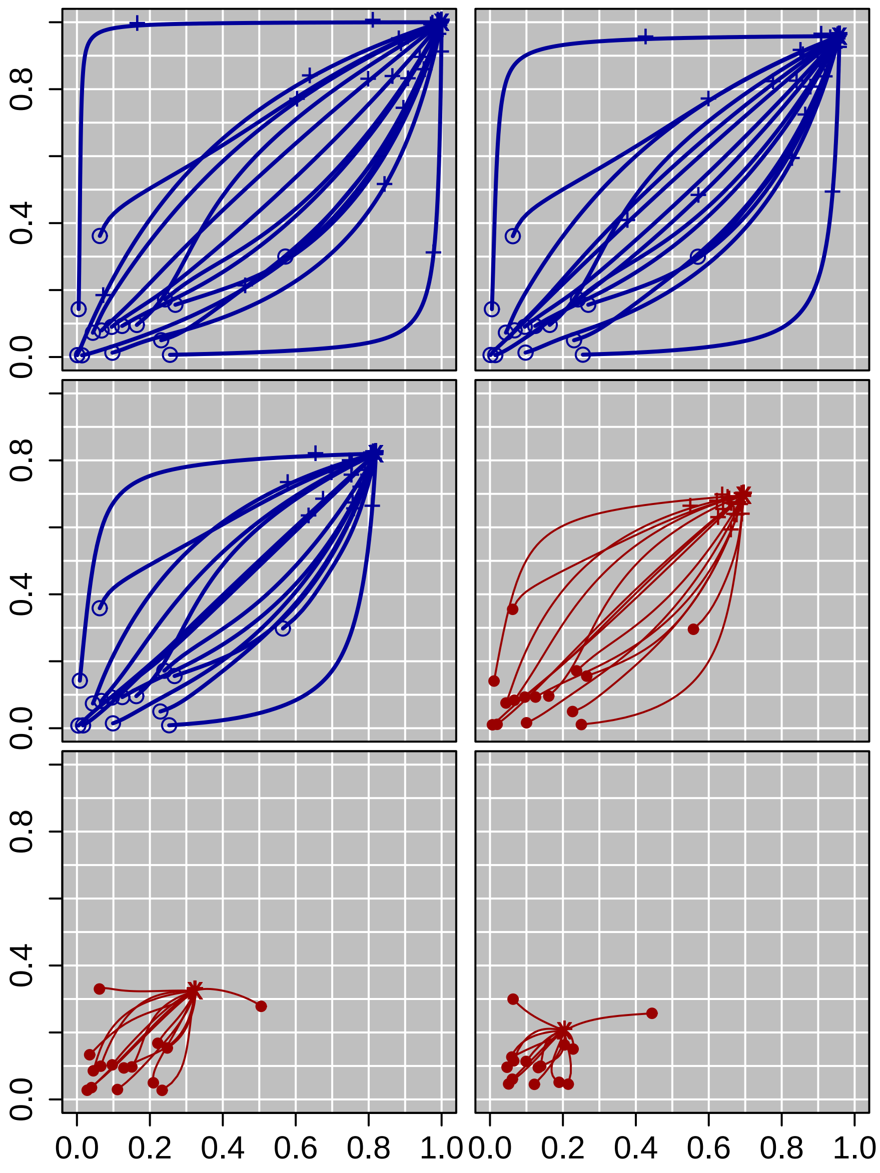


Figure 5.11: 2D takeover plots for MPS trajectories in with bit-flip mutation. Reading from the left to the right, top to bottom $p_m = \{0.00, 0.001, 0.005, 0.01, 0.05, 0.1\}$.

the rest of the fitness landscape. However, notice that the destabilizing effects of crossover do not seem to occur in the case of mutation.

Mutation and Crossover

Validating and visualizing the effects of MPS models with both crossover and mutation ($p_c \in \{0.0, 0.1, 0.2\}$ and $p_m \in \{0.000, 0.005, 0.05\}$) is now a straightforward exercise in producing a combined mixing matrix. It is clear that the basis vectors are no longer fixed points. What is unclear is whether the combination of both operators will amplify the trajectory distortions produced individually, diminish them, or produce some other effect.

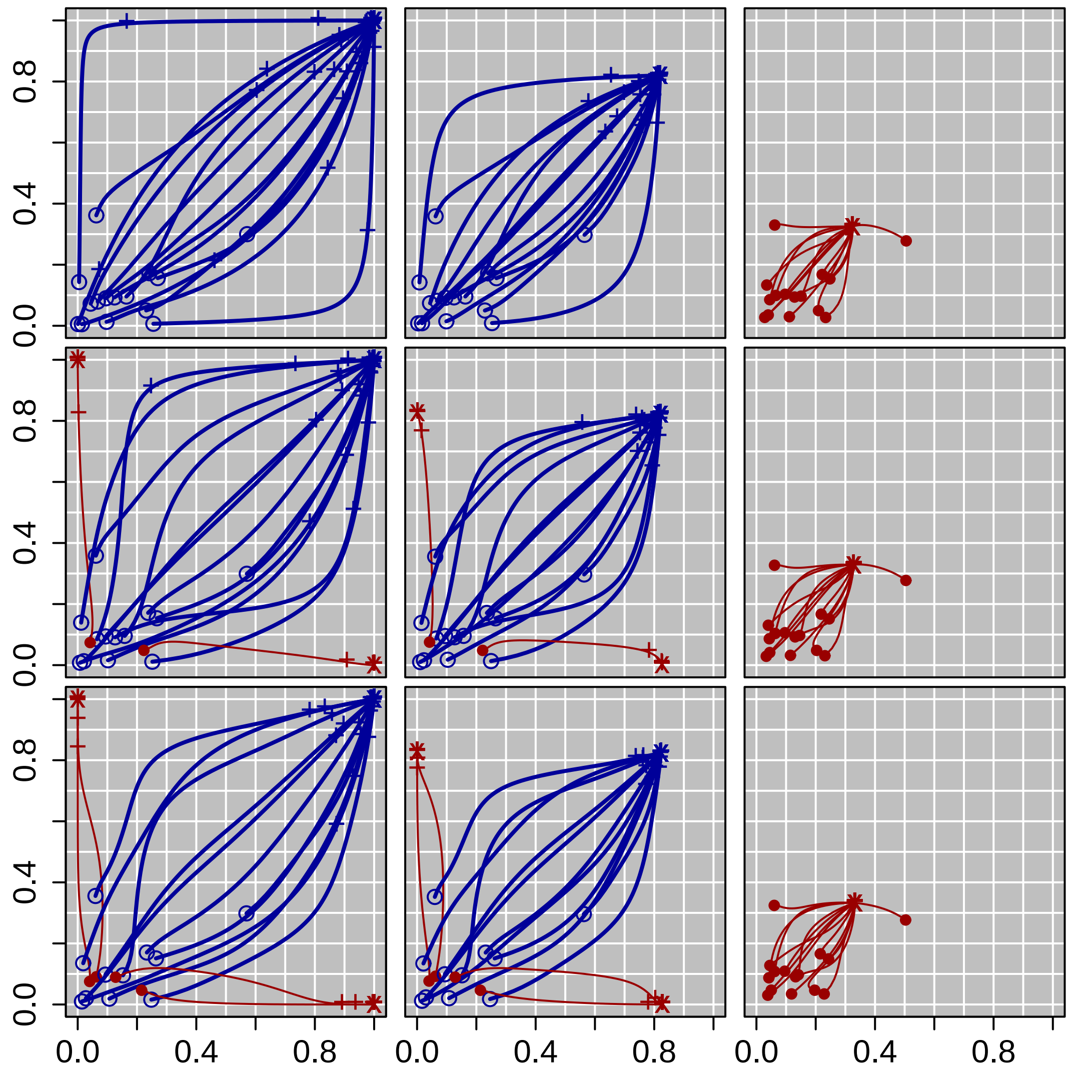


Figure 5.12: 2D takeover plots for MPS trajectories in with bit-flip mutation and parameterized uniform crossover. Reading from the left to right $p_c \in \{0.0, 0.1, 0.2\}$, from top top bottom $p_m \in \{0.000, 0.005, 0.05\}$.

Figure 5.12 on page 90 shows six combinations of the model. Interestingly, the result of combining the variation operators is additive in a sense. In these studies, trajectories fall towards the basis vectors associated with the center four payoff values as they did under crossover alone. The bottom two graphs in figure 5.12 shows a projected version of this effect. However, observing the right two graphs, the limit point is pulled into the interior of the simplex product space due to mutation.

5.3.3 Alternative Mixing Models

One reasonable complaint about the traditional EGT model is its supposition of complete mixing. Real CCEAs rarely evaluate an individual by pairing it with all possible collaborators. Indeed, as I discussed in the previous chapter, many CCEAs use a very few, fixed number of collaborators.

Still, there is no reason that the model cannot be modified to consider alternative mixing models, though what kind of quality of analysis is possible with them is as yet unclear. I will discuss two very simple partial mixing models as a proof of concept. These two methods consider both the idea of biasing the collaboration sampling process an the idea of enforcing a fixed size to the sample: complete-weighted collaboration and *c*-random collaboration.

Complete-Weighted Collaboration

The idea behind complete-weighted collaboration is simple: individuals are paired with all possible individuals from the alternate population, but the degree of contribution to the fitness assessment process is weighted by fitness result of the individual in the previous generation. Implementing this approach requires modifications of the original replicator equations that consider a time variable, t. Given that $t := 1$ is defined in the initial conditions, then the system can be described by the following equations.

$$
\begin{aligned}
u_i(t) &= (A\vec{y})_i \cdot w_i(t-1) && (5.27)\\
w_i(t) &= \left(A^T\vec{x}\right)_i \cdot u_i(t-1) && (5.28)\\
x_i' &= \left(\frac{u_i}{\vec{x}\cdot\vec{u}}\right) x_i && (5.29)\\
y_i' &= \left(\frac{w_i}{\vec{y}\cdot\vec{w}}\right) y_i && (5.30)\\
u_i(0) &= 1\ \forall i \in \{1,\ldots,r\} && (5.31)\\
w_i(0) &= 1\ \forall i \in \{1,\ldots,r\} && (5.32)
\end{aligned}
$$

c-Random Collaboration

Constructing a partial mixing model that uses only a finite number of collaborators in an infinite population is somewhat more difficult. I begin by noting that the original replicator equations (Eq. 5.3–5.6) can be used as is for a single-random collaborator. In such a case, the vector components in the first two utility assessment equations refer to the probability that a particular collaborator is picked. Since the population is infinite, and the collaborator is picked uniformly at random, the result of the same set of equations for complete mixing can be see as a model for the *expected* outcome when one selects a single random collaborator.

To c such selections, return to the idea of using masks. Define a masking bit string of length n and a particular string indicating the outcome of a particular collaboration event in terms of which genotypes will serve as collaborator. Given a distribution of masks, C, the augmented replicator equations follow.

$$\vec{u} = \sum_{j\in\Omega} C_j \cdot A\left(\mathrm{diag}(j)\cdot\vec{y}\right) \tag{5.33}$$

$$\vec{w} = \sum_{j\in\Omega} C_j \cdot A^T\left(\mathrm{diag}(j)\cdot\vec{x}\right) \tag{5.34}$$

$$x_i' = \left(\frac{u_i}{\vec{x}\cdot\vec{u}}\right) x_i \tag{5.35}$$

$$y_i' = \left(\frac{w_i}{\vec{y}\cdot\vec{w}}\right) y_i \tag{5.36}$$

The function $\mathrm{diag}(j)$ considers the bit string j of length n and constructs an $n \times n$ matrix such that the bit string represents the diagonal of the matrix and all other values are zero.

This is a somewhat general mechanism that allows for many kinds of collaboration methods by detailing different kinds of mask distributions; however, I will concern myself with using only a fixed number of c randomly selected collaborators. The result is again the *expected* outcome of the particular collaboration sampling method. For a fixed number of collaborators, the distribution of collaborator masks can be computed as follows.

$$C_j = \begin{cases} \frac{c!(n-c))!}{n!} & \text{if } \|j\| = c \\ 0 & \text{otherwise} \end{cases} \tag{5.37}$$

This model still suggests an averaging of multiple function evaluations is used for credit assignment purposes (i.e., the *hedge* method). There are obvious augmentations that can be applied to construct a model that assigns fitness using the maximum possible payoff of the collaborators, but they have the distinct analytic disadvantage of being discontinuous. Additionally, the two suggestions here may be naturally combined with one another to produced a biased, finite sample expected outcome model for collaboration.

5.4 Empirical Examples

Some empirical examples are helpful since the theoretical models I've provided differ in several significant ways from real CCEAs applied to static optimization problems. One difference is that populations are quite obviously not infinite in size. Another difference is that collaboration methods do not really resemble complete mixing, in general.

However, despite its limitations, this dynamical systems model has provided some concrete answers to relatively philosophical questions, as well as some insight into what some coevolution pathologies might actually look like. Moreover, the infinite population model can be seen as an expectation for the behavior of finite population models. With this, we have an answer to the fundamental question: one cannot *expect* these systems to gravitate toward optimal collaborations, even in the ideal case where stochastic effects are minimal, or non-existent. This corresponds exactly with the relative overgeneralization pathology discussed above.

Moreover, visualization of the model has revealed insight into how asymmetries in initial conditions and problem characteristics can exacerbate asymmetries in rates of evolutionary changes between populations. As such, we can *see* loss of gradient actually occur.

Nevertheless, a bridge between the abstract model in which these pathologies gained clarity, and real algorithms in which they are manifested is necessary. Indeed, now that a better understanding of these two difficulties has been achieved, it is a simple matter to create examples that illustrate them in more realistic settings. This section provides two such examples. In the first case, the pathology of relative overgeneralization is demonstrated using the MAXOFTWOQUADRATICS problem already defined, while in the second case a variant of this problem is used to show loss of gradient. Such examples will make tangible connections between theory and practice.

In both cases I used a two population, generational cooperative coevolutionary algorithm with fitness proportionate selection. Collaboration involved all members of the cooperating population, and the mean result (hedge) was assigned as fitness. This is consistent with the complete mixing idea in the formal model. The previous generation of the alternate population was used to assess fitness for individuals (i.e., the parallel updating mechanism). Each population contained 100 individuals. Individuals are encoded using a binary representation (64 bits), and the genetic operators were bit-flip mutation ($p_m = 1/64$) and parameterized uniform crossover (parameter values vary between experiments, see subsections below). In all, the real CCEA was quite similar to the theoretical model.

Comparisons to a traditional EA are made in both examples. The EA in question is as analogous to the CCEA with which it is compared as is possible. The population contained 200 individuals, and individuals were bit strings of length 128. The same genetic operators were applied, but the mutation rate was set at $1/128$. Again, fitness proportionate selection was used.

5.4.1 Relative Overgeneralization

In order to demonstrate the relative overgeneralization pathology in an empirical setting, the algorithms described above were applied to the MaxOfTwoQuadratics problems defined the parameters listed in Table 5.4.1. Each algorithm was run 50 independent times against each of these 8 problems for 100,000 function evaluations per trial. Crossover was not used in these cases ($p_c = 0.0$).

Table 5.4: This table describes the parameter settings used for the MaxOfTwoQuadratics function in the empirical experiments.

Parameter	**Value(s)**
k_1	150
k_2	140
s_1	$\{2, 4, 8, 16, 32, 64, 128, 256\}$
s_2	1
$(\bar{x}_1, \bar{y}_1)$	(8,1)
$(\bar{x}_2, \bar{y}_2)$	(1,8)

Two graphs were produced to illustrate the results, shown on page 94. The first (on top) shows the averages and confidence intervals for the final best-ever fitness values obtained for each group, while the second more informatively shows the ratio of those values that exceed 140. The purpose of this second plot is to illustrate the ratio trials that escaped the suboptimal peak.

Pairwise t-tests with Bonferoni adjustments show that the best-ever fitness values obtained by the CCEA were statistically inferior to the comparable EA for all groups. In both cases, however, the best-ever result unsurprisingly declines as the value of s_1 increases. From the ratios we see what differs between these two algorithms: in the CCEA case, an increasing number of trials are being mislead and converge to the suboptimal peak, rather than the global optimum. Conversely, the EA does not seem to suffer from this problem as deeply. While average best-ever fitness values are clearly affected by the parameter change, no such drastic increase in the local convergence is observed at very high values of s_1, $s_1 = 256$ in particular (where *none* of the CCEA trials escape local convergence).

5.4.2 Loss of Gradient

The loss of gradient challenge is more difficult to illustrate in realistic settings. To try to do so, let's consider the algorithms described above were applied to a modified form of the MaxOfTwoQuadratics problem class, called AsymmetricTwoQuadratics. In this case, the peak widths can be varied independently to

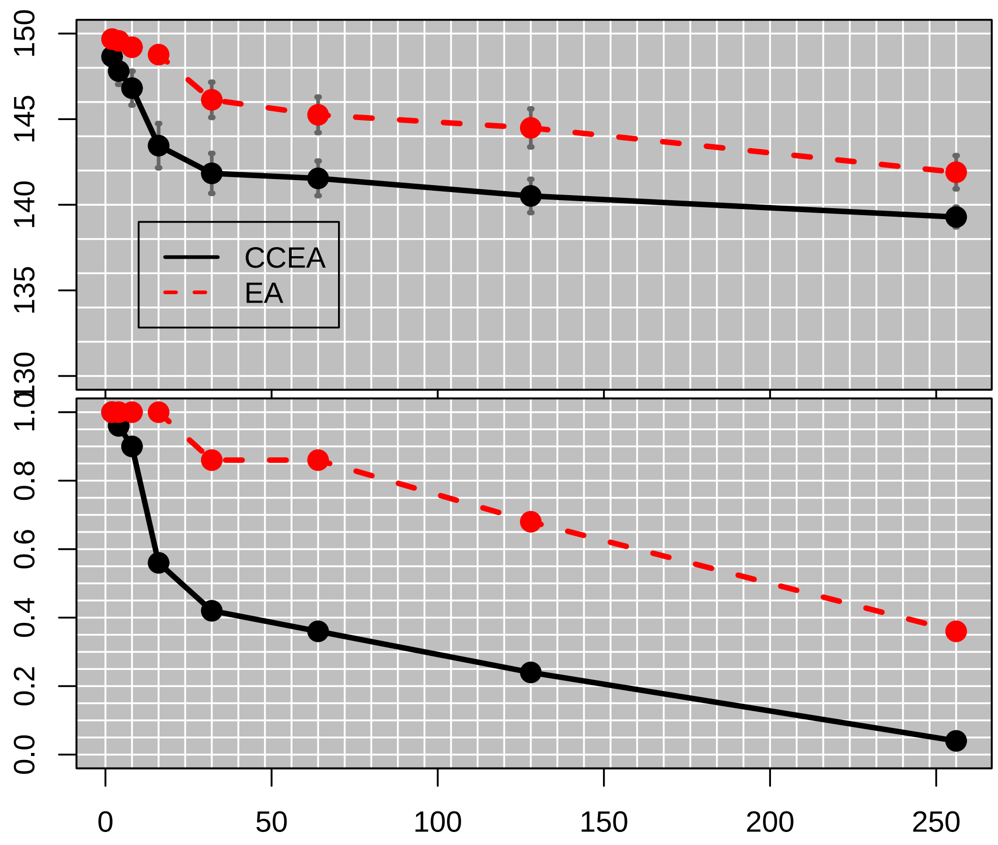


Figure 5.13: CCEA and EA empirical results on MAXOFTWOQUADRATICS as s_1 is increased. The x-axis for both graphs shows the value for the s_1 parameter. In the top graph, the y-axis displays final best-ever fitness results. The points are mean values of 50 trials, while the wings are 95% confidence intervals. The bottom graph shows the ratio of the 50 trials in which the best-ever value exceeds 140.0.

adjust asymmetric conditions in the landscape between the two arguments. The function is defined below. In the experimental groups here, all problem class parameters are defined as they are above, save for the following differences. The s_{x2} and s_{y2} parameters were both set to 1, and the s_{x1} parameter was held fixed at 4. The s_{y1} parameter, however, was varied using the values $\{2,4,8,16,32,64,128,256\}$.

Definition 31. *Given constant values defining the two peaks: k_1 and k_2 to define the peak heights. Asymmetric peak widths are defined by the parameters s_{x1}, s_{x2}, s_{y1}, and s_y2. Again, the points $(\bar{x}_1,\bar{y}_1)$ and $(\bar{x}_2,\bar{y}_2)$ define the locations of the peaks. The function* ASYMMETRICTWOQUADRATICS $:\mathbb{N}\times\mathbb{N}\to\mathbb{R}$ *is defined by*

$$\begin{aligned}
quad_1(x,y) &= k_1 - \left[s_{x1}\cdot(\bar{x}_1 - x)^2 + s_{y1}\cdot(\bar{y}_1 - y)^2\right]\\
quad_2(x,y) &= k_2 - \left[s_{x2}\cdot(\bar{x}_2 - x)^2 + s_{y2}\cdot(\bar{y}_2 - y)^2\right]\\
\textsc{AsymmetricTwoQuadratics}(x,y) &= \max\left(quad_1, quad_2\right)
\end{aligned}$$

Again, each algorithm was run 50 independent times against each of these 8 problems for 100,000 function evaluations apiece. This time crossover was applied ($p_c = 1.0,\ p_s = 0.5$) in order to exaggerate the effects of loss of gradient. The first figure below again shows the best-ever fitness values and convergence ratios for each of the 8 groups with crossover.

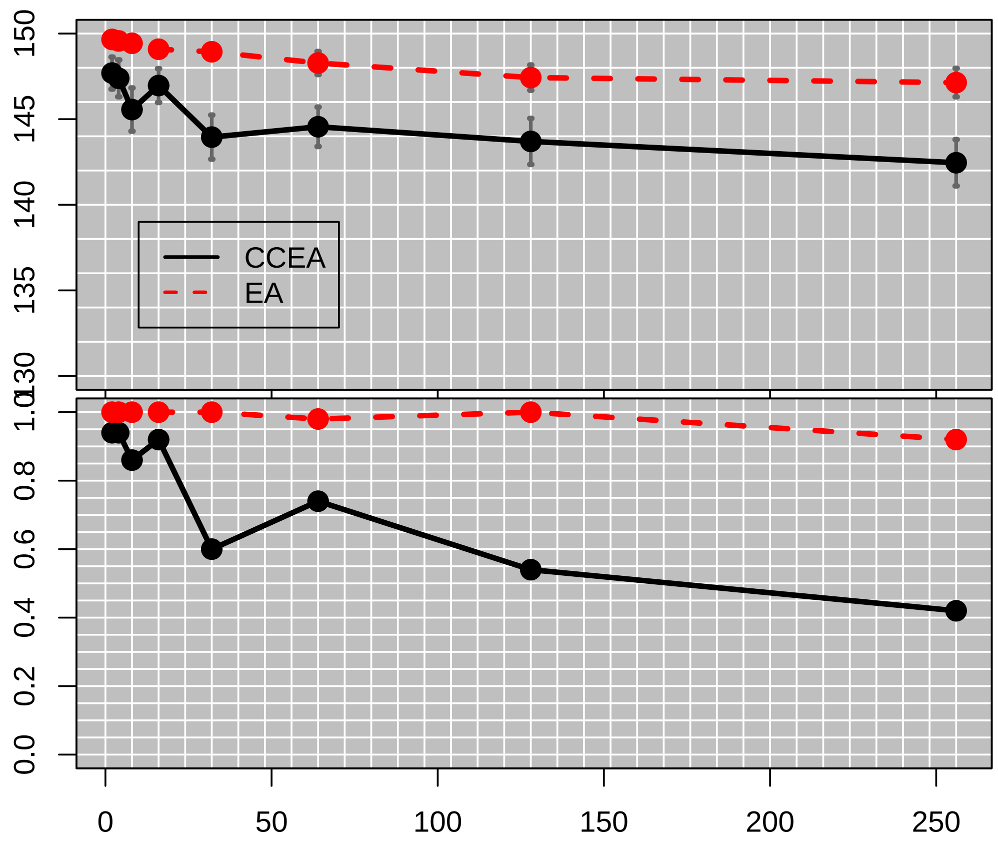


Figure 5.14: CCEA and EA empirical results on ASYMMETRICTWOQUADRATICS without crossover as s_{x1} is increased. The x-axis for both graphs shows the value for the s_{x1} parameter. The y-axis displays final best-ever fitness results. The points are mean values of 50 trials, while the wings are 95% confidence intervals.

Statistically speaking, there is no doubt that the parameter value has an effect on CCEA performance, much as it did in the previous section with the MAXOFTWOQUADRATICS problem. However, with this visualization it is nearly impossible to see what exactly is happening in terms of gradient loss. To see this, I tracked the standard deviation in fitness values for each of the two populations. The resulting standard deviations from the final generation of the two populations can be visualized as a scatter plot of the 50 order pairs considered. The result (shown in Figure 5.15 on page 96) shows that as s_{x1} increases, the it becomes harder and harder for the algorithm to maintain a consistent standard deviation between the two populations. To help see this effect, I've provided a linear regression of the scatter plot. This line "tips" as the landscape becomes more skewed, showing a greater tendency for the populations to attain different levels of diversity. Points begin to line up in the scatter plots, suggesting that when the CCEA terminated, it was typical for one population to be much more diverse than the other.

These graphs indicate loss of gradient. Adding asymmetry causes population diversity between the two populations to become increasingly more asymmetric, the result of which is shown in Figure 5.14. Further, the convergence problems (as illustrated by the ratio graph) seem even more disparate between the EA and CCEA than they did in the MAXOFTWOQUADRATICS case. Indeed, loss of gradient can be among the most serious stumbling blocks for cooperative coevolution.

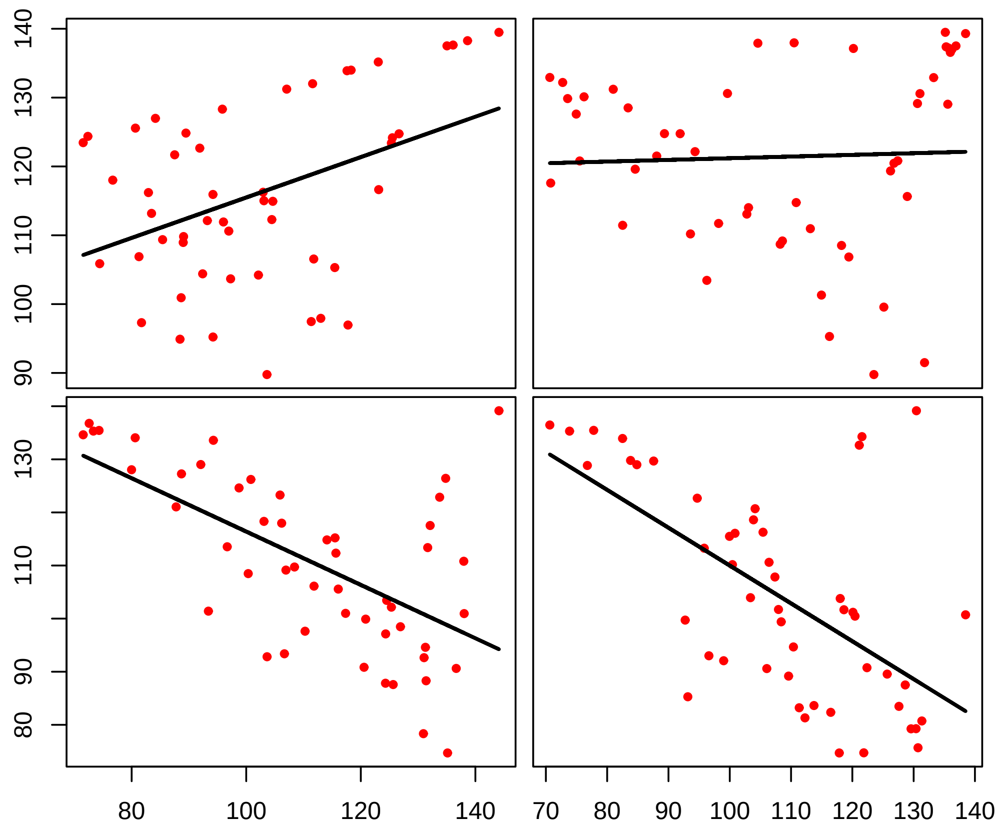


Figure 5.15: These four graphs show co-plotted standard deviations of population fitness at the end of the run. The line is a linear regression fit of the scatter plot. The groups shown appear left to right, top to bottom as $s_{x1} = 4,\ 16,\ 64,$ and 256, respectively.

Chapter 6

New Views of CCEAs

In the previous chapter, I showed that the cooperative coevolutionary algorithm may not be particularly well-suited (in general) to static optimization problems that are straightforwardly encoded. In some sense, this can be seen as the main message of this dissertation. It addresses the fundamental question posed in the introduction: "What does this tool *do*?". Put more specifically, I asked "Do CCEAs optimize?". The answer here is: the tool is *not* a static optimizer of ideal collaboration (i.e., the specific collaboration that results in the optimal fitness), but rather it seems to be an adaptive optimizer of the joint distribution of rewards described by collaboration space.

I believe this knowledge is helpful and useful to practitioners. The general contribution of this is to illustrate plainly to those studying cooperative coevolutionary algorithms that a *new view* of what CCEAs do, and how they work is imperative if we are to make progress in applying them successfully. This chapter attempts to make my main message clearer by offering several specific high level suggestions of such new views. These views stem from answers to the question, "If one should not apply the simple CCEAs described here to straightforward static optimization problems, what is to be done with them?"

There are several things this thesis does *not* say. First, it does not suggest that CCEAs *cannot* be applied to static optimization problems, nor does it even suggest that such applications will never be successful. Second, it does not really directly answer the most basic form of the fundamental question since it is not at all clear what the application-oriented meaning of an "adaptive optimizer of the joint distribution of rewards described by collaboration space" is. These are fair observations, and I will respond to both of these points at the end of the chapter.

This chapter provides the community with some high level advice for the direction that future researchers might take, given the message established here. I take the very simple tack that if the CCEA should not necessarily be expected to perform static optimization tasks, something should be done to either change the existing algorithms or how, and to what, one applies them. With this view in mind, the chapter offers three simple high level suggestions for how one might improve the results of the algorithm by either altering the algorithm itself or by applying it more appropriately. The first section addresses the difficulties caused by the algorithms' propensity towards robust resting balance by describing a simple, but extensible method for biasing the algorithm toward a preference for ideal collaborations. Maintenance of the adaptive evolutionary balance is the subject of the second section, which describes an attempt reduce the problems of gradient loss by augmenting a CCEA with a means of balancing the rates of evolutionary change in the respective populations. The third section provides a short philosophical discussion about the kinds of problems for which traditional cooperative coevolutionary algorithms may be more appropriate. In the final section I will provide a brief conclusion about the main theme of the dissertation.

6.1 Biasing Towards Static Optimization

Part of the difficulty the CCEAs I've discussed thus far face when applied to static optimization problem is the fact that an individual's fitness is commonly assessed only by a subset of the potential interaction space. That is, it is assessed (typically) based on how well it performs with immediate individuals from the other populations.

In a very literal sense, this is taking a *projection* of the total interaction space. Such a projection, generated by sampling the space using the collaborating populations, may or may not provide the search with adequate or appropriate information to lead it towards the ideal collaboration.

This difficulty speaks to the first form of balance discussed earlier, the one most stressed by this dissertation: the tendency toward *robust resting balance*. That is CCEAs tend toward the resting balance of Nash equilibria associated with high joint distribution values in the payoff matrix, not toward optimal collaboration. Since for many very realistic problems such equilibria may be highly-suboptimal, this tendency can lead to one of the pathologies discussed in the previous section: relative overgeneralization.

To find optimal collaborations, the search process may need to be more aggressive than this: assessing fitness based more on the *highest-reward* interactions between an individual and various members of the other population. Indeed, this is part of the spirit of the empirical ideas discussed in Chapter 4 with respect to the use of the optimistic credit assignment collaboration method, which tended to yield better results than when using the mean or minimum performance. This has the advantage of being more opportunistic, forming a projection that considers the basic intent of the optimization algorithm: find the ideal. But it is still a projection—and the information used to construct this projection is still limited to the current context.

The idea presented here is similar to this opportunistic approach, and bears a slight resemblance to the "Hall of Fame" concept introduced by Rosin and Belew (1997) for competitive coevolution. An individual's fitness might be based on a combination of its immediate reward while interacting with individuals in the population and an estimate for the reward it would have received had it interacted with its "ideal collaborators". Moreover, the fraction of reward due to the immediate (as opposed to the ideal) interaction might be adjusted during the course of the run. In this way, the projection that is formed during one part of the search is one that considers a wider context that allows for more exploration of unseen parts of the space, or it may consider a more narrow scope at other times during the search.

This idea of an optimization biased CCEA can be justified in the following way. Recall that if an individual's fitness is based on its immediate interaction with individuals from the other population, then $\vec{u} = A\vec{y}$ and $\vec{w} = A^T\vec{x}$, as described in equations 5.3 and 5.4. Now, let us consider a function $\max(A)$ that returns a column vector corresponding to the maximum value of each row in matrix A. Now, if an individual's fitness is based on its maximum possible performance in conjunction with any individual from the other population, then one may modify equations 5.3 and 5.4 to be $\vec{u} = \max(A)^T$ and $\vec{w} = \max\left(A^T\right)^T$.

In this modified system, the tendency to optimize performance is clear since the new fitness measure will result in populations that converge to basis vectors associated with any unique maximal value in the payoff matrix. At each iteration of the model, the fitness of each strategy will be its best possible fitness. If there is a unique maximum, that result will have the highest fitness, and so the proportion of the corresponding strategy will increase in the next step. The reason for this should be obvious: the problem has lost the dimensionality added due to the nature of the interactions between the populations and has been reduced to a simple evolutionary algorithm. Regardless of the content of the opposing population, the fitness measure for a given strategy is the same. As shown in Reeves and Rowe (2002, Vose (1999), an infinite population model of this reduced evolutionary algorithm will converge to a unique global maximum. In fact, the algorithm is no longer coevolutionary in any real way.

Of course, this idyllic discussion has a major flaw: a real CCEA algorithm would not know the maximum possible reward for a given individual *a priori*. To make use of any bias, this knowledge will have to estimated using some kind of learning mechanism. One approach is to use historical information during the run to approximate the maximum possible collaborative fitness for an individual. However, if the approximation is too large (or has too strong an effect on the overall fitness), and if it appears too early in the evolutionary run, then it can deform the search space to drive search trajectories into suboptimal parts of the space from which they cannot escape. On the other hand, if the approximation affects the fitness measurement very weakly, and too late in the run, then it may not be of much help, and the system will still gravitate towards balance. To better see this tradeoff, equations 5.3 and 5.4 may be again altered such that a bias weight parameter, δ, is added (shown below). Varying δ between 0 and 1 will control the degree to which the model makes use of the bias.

$$\vec{u} = (1-\delta)\cdot A\vec{y} + \delta \cdot \max(A)^T \quad (6.1)$$

$$\vec{w} = (1-\delta)\cdot A^T\vec{x} + \delta \cdot \max\left(A^T\right)^T \quad (6.2)$$

Even in this formal setting there are challenges. The theoretical model itself is enough to see how sensitive the algorithm can be to the δ parameter. To help see this, let's again return to the technique of measuring the basins of attraction discussed in the last chapter. Although this time I will keep track of only the ratio of trajectories that eventually map to the global maximum, the idea is the same: select initial points at random from the product simplex, iterate the model until it has converged to a fixed point, then record whether or not the resting point corresponds with the global maximum. I again consider the MAXOFTWOQUADRATICS problem class as an 8×8 payoff matrix and use the specific problems given by $s_1 = \{2,8,32\}$. Figure 6.1 below shows what happens to the expectation of global convergence as the delta value is increased.

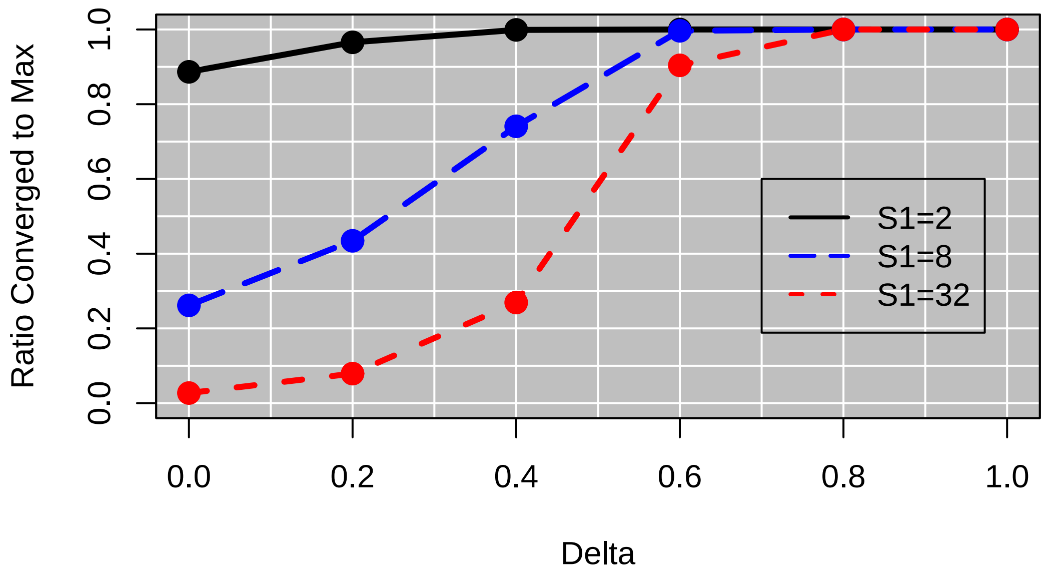


Figure 6.1: Measures of the basin of attraction of the global peak for MAXOFTWOQUADRATICS with varying values of s_1 in terms of the δ bias parameter.

Notice from this figure that as the peak shape is narrowed (as s_1 increases), the sensitivity of the delta parameter is increased. While it is clear that even if we had a reasonably accurate way of estimating the ideal collaborator from historical information, the proper setting for the delta parameter may be quite difficult to attain. For some problems there may be a reasonable range of values for which good results are possible (as is the case with MAXOFTWOQUADRATICS when $s_1 = 2$), while for other problems setting δ may be quite tricky (as is the case with MAXOFTWOQUADRATICS when $s_1 = 32$).

This sensitivity notwithstanding, it is clear that some kind of biasing method may help address the inherent tendency of CCEAs to seek balance rather than optimality. A somewhat more realistic example application of this approach can be found in Panait, Wiegand, and Luke (2003). Here it is shown that for a relatively naive mechanism that uses historical information to estimate the ideal collaborator, it is possible to improve upon results produced to an unaltered CCEA. Moreover, recent work in competitive domains show a similar idea with respect to retaining and using historically found Nash equilibria to keep the search on track (Ficici and Pollack 2003).

There are, of course, many open questions about how to apply such a method in a more realistic way; however, given the right estimation methods and parameter settings, biasing coevolution towards optimality might be one effective way of altering the basic cooperative coevolutionary algorithm to more more geared towards the task of static optimization. For these reasons, it certainly merits further study.

6.2 Balancing Evolutionary Change

Like the previous section, this section attempts to resolve the problem with CCEAs by correcting the algorithm. Indeed, both sections consider the ideas of balance, though differently. In the previous section, the idea was to thwart the natural tendency of the CCEA toward robust resting balance by biasing the algorithm toward optimal collaboration. This section, however, attempts to address the issue of dynamic balance by attempting to preserve that balance.

As I described in Chapter 5, coevolutionary algorithms "progress" while remaining in a type of adaptive balance. To lose that balance means that one or more populations will collapse to near homogeneity, and the remaining populations will lose some, or all, of their gradient for search. In the context of the application of these symmetric CCEAs on static optimization problems, it is clear that if one can keep the algorithm progressing in terms of increased payoff values, the optimum will be reached (eventually). So how can this be done?

Recall that the effects of variation (and possibly even stochastic selection) can exacerbate asymmetries in the initial conditions and the problem itself, which can lead to the destabilization in the balance the the search. The result is often some form of loss of gradient. Is there a way to protect the dynamic balance of the search by dampening these effects?

One possible approach might be to localize the operators using some kind of spatial embedding (Sarma 1998). For example, a two-population CCEA might be distributed on two, matched-up grids such that each grid point corresponds to an individual (e.g., see Figure 6.2). Selection is performed locally within some pre-defined neighborhood, and collaboration is similarly constrained to the corresponding neighborhood in the matched grid. Indeed, such coevolutionary algorithms have been shown to work better than corresponding non-spatial CEAs for some, very specific problems (Pagie 1999). The reasoning behind their advantage has, until now, not been explained.

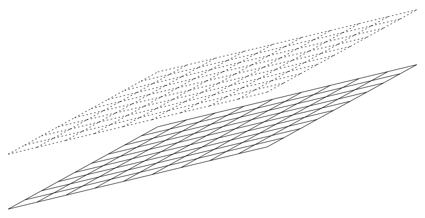


Figure 6.2: Example spatial distribution grids for a two-population CCEA

One reason might be that, transmission of information within a population is not only slowed and localized, but it is in some sense *capped*. That is, there is a maximum computable rate at which information can diffuse across the population, given the relationship of the local neighborhoods to the population itself (Sarma 1998). This means that the two populations might be way "locked" together in terms of their rate of change. Moreover, asymmetries in the problem can be in some way countered (if they are known or suspected *a priori*) by adjusting the neighborhoods for collaborations relative to one another.

To test this idea, the following experiments were performed. A two-population CCEA was applied to the AsymmetricTwoQuadratics static optimization problem class, save that the domain values were scaled between 1.0 and 8.0, as with the experiments discussed at the end of the previous chapter. The parameters to instantiate the problems used here are shown in Table 6.2.

Table 6.1: Parameters for the MaxOfTwoQuadratics problem used in the spatial embedding experiments.

Parameter	**Value**
k_1	150
k_2	140
s_{x1}	$\{2,4,8,16,32,64,128,256\}$
s_{x2}	1
s_{y1}	4
s_{y2}	1
$(\overline{x_1},\overline{y_1})$	(8,1)
$(\overline{x_2},\overline{y_2})$	(1,8)

I compare the CCEA results on the AsymmetricTwoQuadratics problem from the previous chapter to a spatial model where 49 members of each population are laid out in 7×7 toroidal grids. A given grid position refers, then, to two individuals: one in the first population and one in the second. Collaboration consisted of using the 3×3 Moore neighborhood in the alternate population centered at the same position as the individual being evaluated. The mean of these 9 values was used as the fitness value for the individual being evaluated. Proportionate selection likewise used a 3×3 Moore neighborhood around the present individual in the populations. This meant that the 9 individuals associated with a given position were used to construct a weighted distribution, a draw was taken from that distribution, and the winner was substituted into the given position in the next round. The positions were processed in random order (without replacement). Again, crossover was applied as it was before ($p_c = 1.0$ and $p_s = 0.5$). The update mechanism was parallel.

Each group was run 50 times and the best result found during the search was preserved in each trial. The best-ever results, as well as the ratio of final generation best values exceeding 140, are shown in Figure 6.3. The values for every group are far improved over the non-spatial results shown on page 95. The best-ever results are statistically significant for all values of s_{x1}. Looking again at the scatter plots of the standard deviations of the final generation of the runs (Figure 6.4 on page 103) provides with some clue as to why. Though at first it may appear that all of the regression lines are tipped in this case, there are two important things to note. First, the overall standard deviations are much lower than those shown on page 96. Second, the differences between the relationships of standard deviations are much less pronounced between different values of s_{x1}. This suggests that increasing the value of s_{x1}, in effect skewing the problem more and more, had little impact on the relationship of the final generation's population diversities.

Again, it is obvious that there are more questions about how to go about applying this method than there is advice provided here. The point was not to demonstrate a particular method, but to underscore the idea that by preserving the dynamic balance of the CCEA, better results might be obtained by reducing the risks associated with the problem of loss of gradient. Many other methods exist for accomplishing something similar.

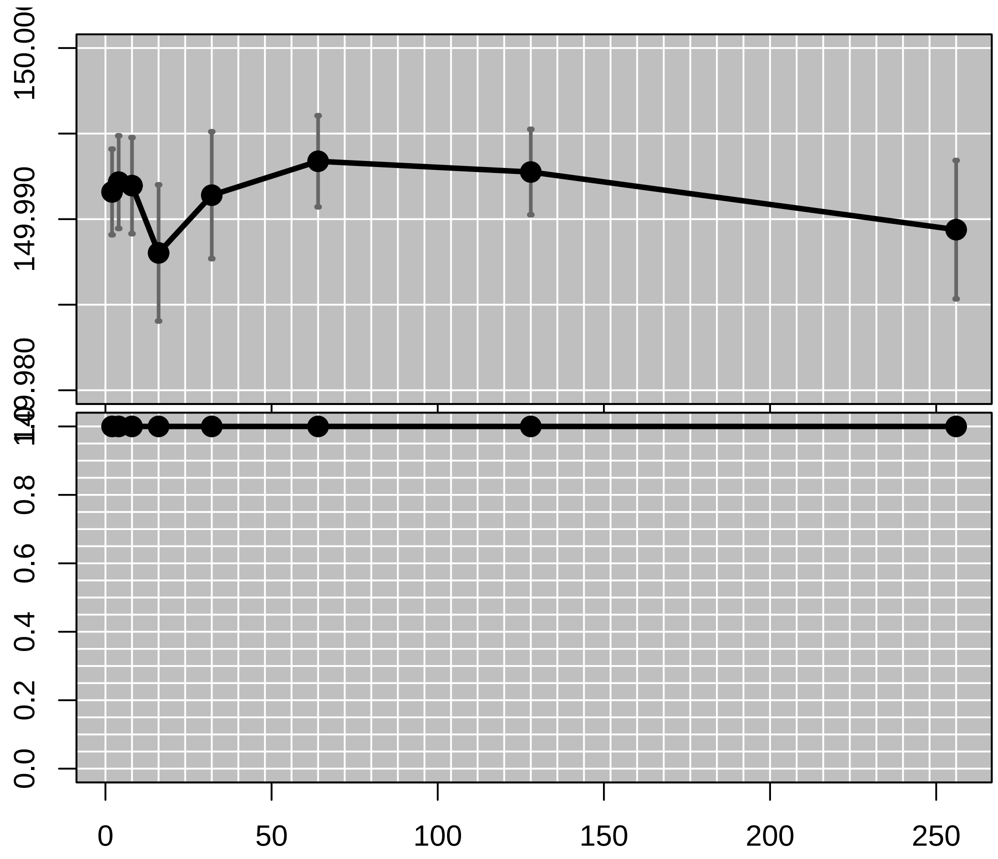


Figure 6.3: Spatial CCEA empirical results on ASYMMETRICTWOQUADRATICS with crossover as s_{x1} is increased. The x-axis for both graphs shows the value for the s_{x1} parameter. The y-axis displays final best-ever fitness results. The points are mean values of 50 trials, while the wings are 95% confidence intervals.

6.3 Optimizing for Robustness

To the last two sections, one might respond with questions like, "Why change the algorithm? Why not apply the existing algorithm to more appropriate problems?" Unfortunately it is not altogether clear what "more appropriate" problems are. Returning to some of the initial semantic discussion may help shed some light on the question, though. Recall that in Chapter 3 I defined the term *single objective, static optimization problem* in such a way as to direct our discussions throughout the rest of the dissertation. Directing it in this way stipulated that "optimal" meant "ideal collaboration" in the simple encoding I chose for the CCEA in this work. Suppose now that the problem being solved is not one in which the ideal collaboration is desired at all, but rather we *want* to find a reasonably good solution that works well across a wide range of alternative collaborations?

Indeed, according to the analysis in the last chapter, this is exactly what CCEAs are constructed to do: find the pure Nash equilibrium with the highest cumulative joint distribution in the payoff matrix. Since the solution is a Nash point, if one strategy in the collaboration is fixed, no alteration in the other strategies will produce a higher reward (such is the definition of a Nash equilibrium). In this sense, the fixed point might be considered "relatively good". Moreover, Nash points with higher joint distributions have greater attracting power, the attracting fixed point is likely to be one in which altering one of the non-fixed strategies will be less damaging than some other Nash point with a lower joint distribution. In this way, we might consider our

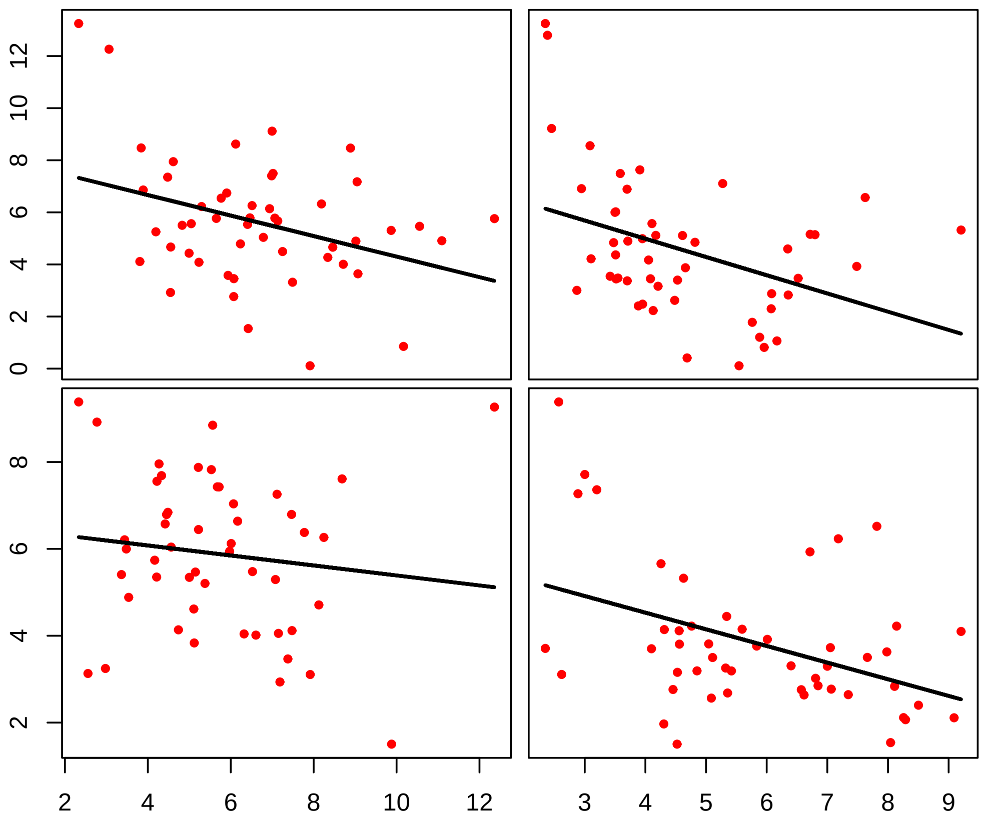


Figure 6.4: These four graphs show co-plotted standard deviations of population fitness at the end of the run. The line is a linear regression fit of the scatter plot. The groups shown appear left to right, top to bottom as $s_{x1} = 4$, 16, 64, and 256, respectively.

solution *robust*: it is relatively good and is susceptible to harm by minor changes in the alternate strategies than other such points. Suppose, then, one is interested in optimizing for robustness, rather than ideal collaboration?

It is reasonable to consider such a viewpoint. Take, for example, the problem of learning behaviors for multiple, collaborating agents in some kind of stochastic domain. Presumably one is, on the one hand, less interested in the best possible solution than a satisfactory one and, on the other hand, more interested in solutions that are reasonably stable in the face of minor fluctuations in the team behaviors. In other words, it is usually better to have a team that will still perform relatively well when one or more members behaves differently than one expects, than it is to have one which performs very well only if every member does exactly what it should.

Consider the MAXOFTWOQUADRATICS, $s_1 = 32$ problem example again. For the sake of argument, let's suppose that this landscape describes the performance results of two, cooperating agents. If the object is to learn good, but robust behaviors, which peak is the "better" peak? The global peak has the disadvantage that any change by either partner in their strategy could bring about a rather dramatic loss of performance. In realistic multi-agent settings, this could happen for a variety of reasons (e.g., a servo on the robot fails for some reason). The second, suboptimal peak offers a much more stable result.

6.4 Conclusions

The main message of this dissertation is a response to the fundamental question posed at its start: one should not expect CCEAs (in general) to perform well on single objective, static optimization problems because dynamically they are predisposed to achieving states of robust resting balance over those of ideal collaboration. Recall, though, that there are at least two important caveats mentioned above: I do not suggest that CCEAs cannot be successfully applied to static optimization problems, nor do I offer a great deal of detailed advice clarifying how they should be applied.

I will now look at these two issues in turn, suggesting future research opportunities as I do so, as well as underscoring the research contributions of this work. Afterward, I will suggest the less tangible methodological contribution of this work.

6.4.1 Applying CCEAs to Static Optimization

Can CCEAs be applied successfully to static optimization problems, despite their internal dynamic tendencies? Of course they can be. In Chapter 4 I show clearly that CCEAs can perform competitively with more traditional EAs on some kinds of problems. Moreover, the No Free Lunch theorem tells us that all algorithms perform (on average) the same against any problem set that is closed under permutation (Schumacher, Vose, and Whitley 2001). One *can* apply CCEAs to such problems, and one *can* expect it to perform reasonably well at times.

Nevertheless, in the more specific situations where practitioners are interested in a subset of problems that have certain, very reasonable and common properties, the use of CCEAs should be reconsidered. I have shown that for very simple problems that contain some kinds of cross-population nonlinearities, the CCEA can have some very pathological difficulties. Moreover, whatever the underlying relevant property that creates challenges for these methods, it is something more complex than simply the existence of such epistatic connections. There are clearly certain *kinds* of nonlinear relationships between populations that create difficulties, and there are clearly other relationships at play as well. The existence of cross-population epistasis is neither a necessary nor sufficient condition to raise the red flag of warning.

However, in spite of the challenges presented by these algorithms, there's some hope that they can be modified to be more suited for static optimization procedures. I suggested two such ways. In the first, the algorithm might be explicitly biased toward ideal collaboration by considering historical information regarding the interaction search space. In the second, I offered one method for how adaptive balance in the algorithm might be preserved by harnessing the constraining effects of spatial embedding as a means of tying changes in two populations together. Both ideas are only one of many ways to address their respective issues. I presented them as conceptual examples, not as specific rigid dictums for how to solve these cooperative coevolutionary difficulties.

At this level of abstraction, these ideas contribute advice to the future of the study of cooperative coevolutionary algorithms in a very real way: researchers should focus on similar kinds of modifications to address these challenges, if they wish to continue to use these algorithms in such settings.

6.4.2 Adaptive optimization of robustness

So what do CCEAs really do? My answer is that they are adaptive optimizers of robustness or, more explicitly, they attempt to adaptively discover places in the search space with maximal joint distribution of rewards as described by the collaboration space. It might be intimated that this is only an indirect answer to the fundamental question since it is not necessarily clear exactly what this statement means. Moreover, while it is important to know what CCEAs *do not* do, it is also important to know what they *do*.

Research into the field of coevolutionary computation has been distracted by several misconceptions about the algorithms themselves, and particularly about how they function. To some extent, we must revisit investigations of CEAs in a more focussed light. The contributions provided here are more than simple disputation because they inform researchers that we must step back and take a new look at the algorithms from a more informed perspective with respect to their function. What is needed is a new view of the CCEA in terms of the problems it solves.

While the clarity of what the real-world implications of "adaptive optimization of robustness" may be in doubt, researchers who view CCEAs through this lens will be more successful at determining how CCEAs can be best applied. If nothing else, future researchers of coevolutionary computation should consider this before beginning their research: make certain that the algorithm is solving the problem that is being encoded, and not something quite different than one might have expected.

6.4.3 A Methodological Framework

Perhaps a far less tangible contribution of this dissertation can be found in its methodological approach. There are two points in this matter that I would like to underscore.

On a general level, there is at least one classical methodological decision made by this research that is worth pointing out. I began with a question and addressed all analysis and experimentation to that question, or subordinate instantiations of that question. Specifically, I began with a very broad question, and refined it in different ways based on context. These refinements produced more specific questions that were more researchable. The process of starting broadly and refining to specific question is a framework that offers great deal in focussing research that has many angles,.

As an example, the "fundamental question" was just such a broad question. I outlined this at the start of the document and continually refined it in each context of my analysis. Moreover, I kept the question as simple as possible. Indeed, I offer this suggestion for future research in coevolutionary computation: continue to *begin* research with that fundamental question: "What do these algorithms really do?"

However, beyond the more obvious and classical methodological observations, I want to say a few words about the analytical tools used in my research. There is a great deal of controversy and tension in the evolutionary computation field about the utility and informativeness of various theoretical ideas. Some believe that dynamical systems approaches tell us very little about what real algorithms will do in terms of how long we can expect to wait for "good" solutions, while others believe that they help us model and understand algorithm limiting behaviors as they are in their most abstract and simplest form. Some believe that run time analysis is unhelpful because it can only answer questions about very simple, unrealistic algorithms on very contrived problems, while others believe that they provide very specific and precise information about real algorithms (rather than models of algorithms). Even more extremely, some believe that no theoretical approach has so far proved helpful to practitioners, while others hold practitioners as irresponsible for poor empiricism born out of a lack of fundamental understanding of the algorithms they use.

What is the truth? The truth is that one needn't discard any of these tools, but recognizing their advantages and limitations, one can apply many them in different ways. *Many* of these methods can help answer *some* types questions, as long as the questions are properly posed. It may be difficult to understand how dynamical systems methods can be used to inform us of how long users must wait for solutions, but it is not difficult to understand how they can inform us about whether not the algorithms are even attracted to those solutions in the first place. It may be that run time analysis is nearly impossible for the complicated algorithms that are actually employed in real-world environments, but it is also true that knowing something about the run time behaviors of simpler algorithms on well constructed problems with useful properties is more than knowing nothing at all in terms of what kind of performance we can expect from similar EAs. Moreover, to fill the gaps left by various formal methods, there is always empiricism. The key to understanding these algorithms is to take a multi-lateral approach in terms of formalism, to use the results from a variety of formal methods to pose real and informative hypotheses for testing, then validate and confirm (or reject) these hypotheses by empirical investigation.

In particular, I believe that formal methods help in the process of refining questions. Transforming broad questions into particular, researchable questions from which hypotheses can be generated is difficult. Here is one place for formalism. Which theory is used says more about *how* one wishes to refine the question than it does about their theoretical philosophy in general. From these more specific questions and hypotheses, one can design and implement experiments that are very compact and focussed. The key message here is that analysis should begin with formalism and end with empiricism, and in both cases researchers should not limit themselves by ascribing to, or omitting, any particular method merely on the basis of its inability to answer other questions that one has. They each have their use, and if applied carefully in the context of a particular question, can be informative in their own way.

6.5 Future Research

There are three areas of research extensions of this work that are foremost on my list of items to pursue in the near future. First, I am interested in continuing the theoretical run time analytic work. I also continue to have an interest in working on biasing methods to improve CCEA optimization applications. Finally, I would like to turn game-theoretic tools toward the task of understanding specialization in multi-agent systems in which behaviors are learned via CCEA methods.

I continue to believe that run time analysis can provide powerful and decisive answers to particular questions with respect to the performance of CCEAs on optimization problems. I am interested in extending the analysis already accomplished in two, very different ways. First, the next step in analysis would seem to be moving to providing theoretical complexity results for a CC $(\mu + 1)$ EA. Such an algorithm is a good next-step, since it is similar to a steady-state GA and constitutes a relatively obvious augmentation of the (1+1) versions discussed in the current work. Second, I would like to study the effects that different collaboration mechanisms can have on run time performance. This can be done even without a population by using an external source to provide collaborators (they can be provided randomly, for instance). Such methods would not be identical to true collaboration, but may still provide useful results.

Finding a biasing mechanism that will help focus the CCEA more on static optimization will be of great value to the community of practitioners. The key lies in developing a realistic and tractable method for learning increasingly more optimal collaborations during the run of the algorithm. Even if such methods could be determined, some problems will make them more suitable than others. Understanding what properties of the problem make this a tenable method and what properties make it infeasible continues to interest me.

As I've already said, I am convinced that the native CCEA's true potential will be found by applying it to problems in which our interest is finding robust solutions, and that one such problem domain might be that of learning multi-agent behaviors. One of the challenges of learning such behaviors is that of determining what kinds of specialization of behaviors is needed within an agent team, as well as what kinds of specialization are even possible. I believe that the game-theoretic and dynamical systems tools established here will provide a good foundation for answering such questions. It is my intention to continue the research started in my dissertation by extending the analytical and empirical methods used here to the domain of multi-agent learning via cooperative coevolution.

BIBLIOGRAPHY

Alligood, K., T. Sauer, and J. Yorke (1996). *Chaos: An Introduction to Dynamical Systems*. Springer–Verlag.

Angeline, P. (1995). Adaptive and self-adaptive evolutionary computations. In *Computational Intelligence: A Dynamic Systems Perspective*, pp. 152–163.

Angeline, P. and J. Pollack (1993). Competitive environments evolve better solutions for complex tasks. In S. Forest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA)*, San Mateo, CA, pp. 264–270. Morgan Kaufmann.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolutionary Straegies, Evolutionary Programming, and Genetic Algorithms*. Oxford Press.

Beyer, H.-G. (2001). *Theory of Evolution Strategies*. Springer-Verlag.

Bucci, A. and J. B. Pollack (2002). Order-theoretic analysis of coevolution problems: Coevolutionary statics fix. See Poli, Rowe, and Jong (2002), pp. ??–??

Bucci, A. and J. B. Pollack (2003). Focusing versus intransitivity geometrical aspects of co-evolution. See Cantú-Paz, Foster, Deb, Davis, Roy, O'Reilly, Beyer, Standish, Kendall, Wilson, Harman, Wegener, Dasgupta, Potter, Schultz, Dowsland, Jonoska, and Miller (2003), pp. 250–261.

Bull, L. (1997). Evolutionary computing in multi-agent environments: Partners. In T. Baeck (Ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA)*, pp. 370–377. Morgan Kaufmann.

Bull, L. (1998). Evolutionary computing in multi-agent environments: Operators. In V. W. Porto, N. Saravanan, G. Waagen, and A. E. Eiben (Eds.), *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, pp. 43–52. Springer-Verlag.

Bull, L. (2001). On coevolutionary genetic algorithms. *Soft Computing 5*, 201–207.

Cantú-Paz, E., J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller (Eds.) (2003). *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2003*, Berlin, Germany. Springer.

Cliff, D. and G. F. Miller (1995). Tracking the red queen: Measurements of adaptive progress in co–evolutionary sumulations. In *Proceedings of the Third European Conference on Artificial Life*, pp. 200–218. Springer–Verlag.

Cormen, T., C. Leiserson, R. Rivest, and C. Stein (2001). *Introduction to Algorithms, Second Edition*. MIT Press.

Davidor, Y. (1990). Epistasis variance: A viewpoint on ga-hardness. See Rawlins (1990), pp. 23–35.

Davidor, Y. and H.-P. Schwefel (Eds.) (1994). *Proceedings of the Third International Conference on Parallel Problem Solving from Nature (PPSN III)*. Springer-Verlag.

Dawkins, R. and J. R. Krebs (1979). Arms races between and within species. *Proceedings of Royal Society of London B 205*, 459–511.

De Jong, K. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ph. D. thesis, University of Michigan, Ann Arbor, MI.

De Jong, K. (to appear in 2004). *Evolutionary Algorithms*. MIT Press.

De Jong, K. and J. Sarma (1992). Generation gaps revisited. In D. Whitley (Ed.), *Foundations of Genetic Algorithms (FOGA) II*, pp. 19–28. Morgan Kaufmann.

Droste, S., T. Jansen, and I. Wegener (2002). On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science 276*, 51–81.

Eiben, A. E., T. Baeck, M. Schoenauer, and H.-P. Schwefel (Eds.) (1998). *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature (PPSN V)*. Springer-Verlag.

Eiben, A. E. and Z. Michalewicz (1998). Parameter control, adaptation, and self-adaptation in evolutionary algorithms. See Eiben, Baeck, Schoenauer, and Schwefel (1998).

Eriksson, R. and B. Olsson (1997). Cooperative coevolution in inventory control optimisation. In G. Smith, N. Steele, and R. Albrecht (Eds.), *Proceedings of the Third International Conference on Artificial Neural Networks and Genetic Algorithms*, University of East Anglia, Norwich, UK. Springer.

Ficici, S. and J. Pollack (1998). Challenges in coevolutionary learning: Arms–race dynamics, open–endedness, and mediocre stable states. In A. et al (Ed.), *Proceedings of the Sixth International Conference on Artificial Life*, Cambridge, MA, pp. 238–247. MIT Press.

Ficici, S. and J. Pollack (2000a). Effects of finite populations on evolutionary stable strategies. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2000*, pp. 880–887. Morgan Kaufmann.

Ficici, S. and J. Pollack (2000b). Game–theoretic investigation of selection methods used in evolutionary algorithms. See Whitley (2000), pp. 880–887.

Ficici, S. and J. Pollack (2000c). A game-theoretic approach to the simple coevolutionary algorithm. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo, and H.-P. Schwefel (Eds.), *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI)*, pp. 467–476. Springer-Verlag.

Ficici, S. and J. Pollack (2001). Pareto optimality in coevolutionary learning. Technical report, Brandeis University.

Ficici, S. and J. Pollack (2003). A game-theoretic memory mechanism for coevolution. See Cantú-Paz, Foster, Deb, Davis, Roy, O'Reilly, Beyer, Standish, Kendall, Wilson, Harman, Wegener, Dasgupta, Potter, Schultz, Dowsland, Jonoska, and Miller (2003), pp. 286–297.

Fogel, D. and G. Fogel (1995). Evolutionary stable strategies are not always stable under evolutionary dynamics. In J. R. McDonnel, R. G. Reynolds, and D. Fogel (Eds.), *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, Cambridge, MA, pp. 565–577. MIT Press.

Fogel, D., G. Fogel, and P. Andrews (1995). On the instability of evolutionary stable strategies. *BioSystems 44*, 135–152.

Fogel, D. B. (1998). Unearthing a fossil from the history of evolutionary computation. *Fundamenta Informaticae 35*(1-4), 1–16.

Fogel, G., P. Andrews, and D. Fogel (1998). On the instability of evolutionary stable strategies in small populations. *Ecological Modeling 109*, 283–294.

Garnier, J., L. Kallel, and M. Schoenauer (1999). Rigorous hitting times for binary mutations. *Evolutionary Computation 7*(2), 173–203.

Goldberg, D. and K. Deb (1990). A comparative analysis of selection schemes used in genetic algorithms. See Rawlins (1990), pp. 69–93.

Goldberg, D. and J. Richardson (1988). Genetic algorithms with sharing for multimodel function optimization. In J. Grefenstette (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms (ICGA)*, Hillsdale, New Jersey, pp. 41–49. Laurence Erlbaum Associates.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison Wesley.

Hancock, G. R. and A. J. Klockars (1996). The question for alpha; developments in multiple comparison procedures in the quarter century since games (1971). *Review of Educational Research 66*, 269–306.

Hillis, D. (1991). Co-evolving parasites improve simulated evolution as an optimization procedure. *Artificial Life II, SFI Studies in the Sciences of Complexity 10*, 313–324.

Hofbauer, J. and K. Sigmund (1998). *Evolutionary Games and Population Dynamics*. Cambridge University Press.

Holland, J. H. (1992). *Adaptation in natural and artificial systems (2nd ed.)*. MIT Press.

Jansen, T. and R. P. Wiegand (2003a). The cooperative coevolutionary (1+1) EA. Technical Report SFB 531, CI 145/03, Universität Dortmund, Dortmund, Germany.

Jansen, T. and R. P. Wiegand (2003b). Exploring the explorative advantage of the copoperative coevolutionary (1+1) EA. See Cantú-Paz, Foster, Deb, Davis, Roy, O'Reilly, Beyer, Standish, Kendall, Wilson, Harman, Wegener, Dasgupta, Potter, Schultz, Dowsland, Jonoska, and Miller (2003), pp. 310–321.

Jansen, T. and R. P. Wiegand (2003c). Sequential versus parallel cooperative coevolutionary (1+1) eas. In B. McKay (Ed.), *Proceedings of CEC 2003*, pp. 30–37. IEEE Press.

Juillé, H. and J. Pollack (1998). Coevolutionary learning: a case study. In *Proceedings of the Fifteenth International Conference on Machine Learning*, Madison, Wisconsin.

Juillé, H. and J. Pollak (1996). Co-evolving interwined spirals. In L. Fogel, P. Angeline, and T. Bäck (Eds.), *Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pp. 461–468. MIT Press.

Kauffman, S. (1991). Coevolution to the edge of chaos: coupled fitness landscapes, poised states, and coevolutionary avalanches. In C. Langton, C. Taylor, J. Farmer, and S. Rasmussen (Eds.), *Artificial Life II: Studies in the Sciences of Complexity*, Volume X, pp. 325–369. Addison-Wesley.

Langdon, W. B., E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska (Eds.) (2002). *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2002*. Morgan Kaufmann.

Liekens, A., H. Eikelder, and P. Hilbers (2003). Finite population models of co-evolution and their application to haploidy versus diploidy. See Cantú-Paz, Foster, Deb, Davis, Roy, O'Reilly, Beyer, Standish, Kendall, Wilson, Harman, Wegener, Dasgupta, Potter, Schultz, Dowsland, Jonoska, and Miller (2003), pp. 344–355.

Luke, S., C. Hohn, J. Farris, G. Jackson, and J. Hendler (1998). Co-evolving soccer softbot team coordination with genetic programming. In *RoboCup-97: Robot Soccer World Cup I (Lecture Notes in AI No. 1395)*, Berlin, pp. 398–411. Springer-Verlag.

Luke, S. and L. Panait (2002a). A comparison of two competitive fitness functions. See Langdon, Cantú-Paz, Mathias, Roy, Davis, Poli, Balakrishnan, Honavar, Rudolph, Wegener, Bull, Potter, Schultz, Miller, Burke, and Jonoska (2002), pp. 503–511.

Luke, S. and L. Panait (2002b). Fighting bloat with nonparametric parsimony pressure. See Merelo Guervós, Adamidis, Beyer, Fernández-Villacañas, and Schwefel (2002), pp. 411–421.

Luke, S. and R. P. Wiegand (2002). Guaranteeing coevolutionary objective measures. See Poli, Rowe, and Jong (2002), pp. 237–251.

Mayer, H. (1998). Symbiotic coevolution of artificial neural networks and training data sets. See Eiben, Baeck, Schoenauer, and Schwefel (1998), pp. 511–520.

Maynard-Smith, J. (1982). *Evolution and the Theory of Games*. Cambridge University Press.

Menon, A. (2002). The point of point crossover: Shuffling to randomness. See Langdon, Cantú-Paz, Mathias, Roy, Davis, Poli, Balakrishnan, Honavar, Rudolph, Wegener, Bull, Potter, Schultz, Miller, Burke, and Jonoska (2002), pp. 463–471.

Merelo Guervós, J. J., P. Adamidis, H.-G. Beyer, J.-L. Fernández-Villacañas, and H.-P. Schwefel (Eds.) (2002). *Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature (PPSN VII)*, Berlin, Germany. Springer-Verlag.

Michalewicz, Z. (1996). *Evolution Programs = Genetic Algorithms + Data Structures*. Springer–Verlag.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.

Moriarty, D. and R. Miikkulainen (1997). Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation 5*(4), 373–399.

Motwani, R. and P. Raghavan (2000). *Randomized Algorithms*. Cambridge Press.

Muehlenbein, H. (1992). How genetic algorithms really work: Mutation and hill-climbing. In R. Maenner and B. Manderick (Eds.), *Proceedings of the Second International Conference on Parallel Problem Solving from Nature (PPSN II)*, pp. 15–26. Springer-Verlag.

Nash, J. (1950). Equilibrium points in n-person games. In *Proceedings of the National Academy of Sciences*.

Olsson, B. (2001). Co-evolutionary search in asymmetric spaces. *Information Sciences 133*, 103–125.

Osborne, M. and A. Rubinstein (1998). *A Course in Game Theory*. The MIT Press.

Pagie, L. (1999). *Information Integration in Evolutionary Processes*. Ph. D. thesis, Universiteit Utrecht, Netherlands.

Pagie, L. and P. Hogeweg (2000). Information integration and red queen dynamics in coevolutionary optimization. See Whitley (2000), pp. 1260–1267.

Pagie, L. and M. Mitchell (2001). A comparison of evolutionary and coevolutionary search. See Spector, Goodman, Wu, Langdon, Voigt, Gen, Sen, Dorigo, Pezeshk, Garzon, and Burke (2001), pp. 20–25.

Panait, L., R. P. Wiegand, and S. Luke (2003). Improving coevolutionary search for optimal multiagent behaviors. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, pp. 653–658. Morgan Kaufmann.

Paredis, J. (1994). Steps towards co-evolutionary classification networks. In R. A. Brooks and P. Maes (Eds.), *Artificial Life IV, Proceedings of the fourth International Workshop on the Synthesis and Simulation of Living Systems.*, pp. 359–365. MIT Press.

Poli, R., J. Rowe, and K. D. Jong (Eds.) (2002). *Foundations of Genetic Algorithms (FOGA) VII*. Morgan Kaufmann.

Pollack, J. and A. Blair (1998). Coevolution in the successful learning of backgammon strategy. *Machine Learning 32*(3), 225–240.

Potter, M. (1997). *The Design and Analysis of a Computational Model of Cooperative CoEvolution*. Ph. D. thesis, George Mason University, Fairfax, Virginia.

Potter, M. and K. De Jong (1994). A cooperative coevolutionary approach to function optimization. See Davidor and Schwefel (1994), pp. 249–257.

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

Price, P. W. (1998). *Biological Evolution*. Saunders College Publishing.

Rabani, Y., Y. Rabinovich, and A. Sinclair (1998). A computational view of population genetic. *Random Structures and Algorithms 12*, 313–334.

Rapoport, A. (1970). *Two-Person Game Theory: The Essential Ideas*. The University of Michigan Press.

Rawlins, G. (Ed.) (1990). *Foundations of Genetic Algorithms (FOGA)*. Morgan Kaufmann.

Rechenberg, I. (1973). *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Frommann-Holzboog.

Reeves, C. (2002). The 'crossover landscape' and the hamming landscape for binary search spaces. See Poli, Rowe, and Jong (2002), pp. 81–97.

Reeves, C. and J. Rowe (2002). *Genetic Algorithms Principles and Perspectives: A Guide to GA Theory*. Kluwer.

Rogers, A. and A. Prügel-Bennett (1998). Modeling the dynamics of a steady-state genetic algorithm. In W. Banzhaf and C. Reeves (Eds.), *Foundations of Genetic Algorithms (FOGA) V*, pp. 57–68. Morgan Kaufmann.

Rosin, C. (1997). *Coevolutionary Search Among Adversaries*. Ph. D. thesis, University of California, San Diego.

Rosin, C. and R. Belew (1995). Methods for competitive co-evolution: Finding opponents worth beating. In L. Eshelman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA)*, pp. 373–380. Morgan Kaufmann.

Rosin, C. and R. Belew (1996). New methods for competitive coevolution. *Evolutionary Computation 5*(1), 1–29.

Rosin, C. and R. Belew (1997). New methods for competitive coevolution. *Evolutionary Computation 5*(1), 1–29.

Rudolph, G. (1997). *Convergence Properties of Evolutionary Algorithms*. Dr. Kovač.

Sarma, J. (1998). *An Analysis of Decentralized and Spatially Distributed Genetic Algorithms*. Ph. D. thesis, George Mason University, Fairfax, Virginia.

Sarma, J. and K. De Jong (1996). An anlysis of the effects of neighborhood size and shape on local selection algorithms. In H. Voigt, W. Ebeling, I. Rechenberg, and H. Schwefel (Eds.), *Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature (PPSN IV)*, pp. 236–244. Springer-Verlag.

Schaffer, J. D. (Ed.) (1989). *Proceedings of the Third International Conference on Genetic Algorithms (ICGA)*. Morgan Kaufmann.

Scharnow, J., K. Tinnefeld, and I. Wegener (2002). Fitness landscapes based on sorting and shortest paths problems. See Merelo Guervós, Adamidis, Beyer, Fernández-Villacañas, and Schwefel (2002), pp. 54–63.

Schlierkamp-Voosen, D. and H. Mühlenbein (1994). Strategy adaptation by competing subpopulations. See Davidor and Schwefel (1994), pp. 199–108.

Schmitt, L. (2001). Theory of GAs. *Theoretical Computer Science 259*, 1–61.

Schmitt, L. (2003). Coevolutionary convergence to global optima. Technical report, The University of Aizu, Aizu-Wakamatisu City, Japan.

Schumacher, C., M. Vose, and D. Whitley (2001). The no free lunch and problem description length. See Spector, Goodman, Wu, Langdon, Voigt, Gen, Sen, Dorigo, Pezeshk, Garzon, and Burke (2001), pp. 565–570.

Schwefel, H. (1995). *Evolution and Optimum Seeking*. New York: John Wiley and Sons.

Sims, K. (1994). Evolving 3D morphology and behavior by competition. In R. A. Brooks and P. Maes (Eds.), *Artificial Life IV, Proceedings of the fourth International Workshop on the Synthesis and Simulation of Living Systems.*, pp. 28–39. MIT Press.

Spears, W. (1994). Simple subpopulation schemes. In *Proceedings of the 1994 Evolutionary Programming Conference*. World Scientific.

Spears, W. and K. De Jong (1991). On the virtues of parameterized uniform crossover. In R. Belew and L. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA)*, pp. 230–236. Morgan Kaufmann.

Spector, L., E. D. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke (Eds.) (2001). *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2001*. Morgan Kaufmann.

Stanley, K. O. and R. Miikkulainen (2002). The dominance tournament method of monitoring progress in coevolution. In *Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2002*.

Stanley, K. O. and R. Miikulainen (2002). Competitive coevolution through evolutionary complexification. Technical Report AI-02-298, University of Texas at Austin, Austin, Texas.

Subbu, R. and A. C. Sanderson (2000). Modeling and convergence analysis of distributed coevolutionary algorithms. See Whitley (2000), pp. 1276–1283.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. See Schaffer (1989), pp. 2–9.

Syswerda, G. (1990). A study of reproduction in generational and steady-state genetic algorithms. See Rawlins (1990), pp. 17–27.

Vose, M. (1999). *The Simple Genetic Algorithm*. MIT Press.

Watson, R. and J. Pollack (2001). Coevolutionary dynamics in a minimal substrate. See Spector, Goodman, Wu, Langdon, Voigt, Gen, Sen, Dorigo, Pezeshk, Garzon, and Burke (2001), pp. 702–709.

Wegener, I. (2002). Methods for the analysis of evolutionary algorithms on pseudo-boolean functions. In R. Sarker, X. Yao, and M. Mohammadian (Eds.), *Evolutionary Optimization*, pp. 349–369. Kluwer.

Weibull, J. (1992). *Evolutionary game theory*. Cambridge, MA: MIT Press.

Whitley, D. (1989). The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. See Schaffer (1989), pp. 116 – 121.

Whitley, D. (Ed.) (2000). *Proceedings of CEC 2000*. IEEE Press.

Wiegand, R. P. (1998). Applying diffusion to a cooperative coevolutionary model. See Eiben, Baeck, Schoenauer, and Schwefel (1998), pp. 560–569.

Wiegand, R. P., W. Liles, and K. De Jong (2001). An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. See Spector, Goodman, Wu, Langdon, Voigt, Gen, Sen, Dorigo, Pezeshk, Garzon, and Burke (2001), pp. 1235–1242.

Wiegand, R. P., W. Liles, and K. De Jong (2002a). Analyzing cooperative coevolution with evolutionary game theory. In D. Fogel (Ed.), *Proceedings of CEC 2002*, pp. 1600–1605. IEEE Press.

Wiegand, R. P., W. Liles, and K. De Jong (2002b). Modeling variation in cooperative coevolution using evolutionary game theory. See Poli, Rowe, and Jong (2002), pp. 231–248.

Wolpert, D. and W. Macready (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation 1*, 67–82.

Zuev, S. and W. Kahan (2001, October). Solutions to problems for math. h90 # 5. http://www.cs.berkeley.edu/∼wkahan/MathH90/S30Oct01.pdf. Internet PDF.

CURRICULUM VITAE

R. Paul Wiegand received a Bachelor of Science degree in Computer Science from Winthrop University in 1994, a Master of Science degree in Computer Science from University of North Carolina Charlotte in 1999, and a Doctor of Philosophy degree from George Mason University in the winter of 2004. His major research interest is in theory of evolutionary computation, coevolutionary computation in particular.

Paul Wiegand has published over a dozen refereed papers in these areas, having presented many of them at major international conferences. He helped organize a coevolution workshop at the Genetic and Evolutionary Computation Conference in the summer of 2002, and served as an invited researcher at Dortmund University as part of the Collaborative Research Center "Computational Intelligence" (SFB 531). He is currently co-moderator of EC Digest, a research-oriented email list that serves the evolutionary computation community and is a member of the IEEE taskforce on coevolution.