CHAPTER 1

Introduction

The natural world exhibits startling complexity and richness at all scales. Examples include complex social systems, immune and nervous systems, and the intricate interrelationships between species. These are but a few of the wonders that have become more apparent as we have increased our ability to examine ourselves and the world around us. Science is an ever-changing system of beliefs that attempts to account for what we observe, and it has adjusted itself dramatically to accommodate incoming data. It is remarkable that so much of what we see has come to be explained by a single theory: the theory of evolution through variation and selection.

Evolution through natural selection has profoundly affected our view of the world. The society into which Charles Darwin delivered what has become known as *The Origin of Species* in 1859 was primed for this change [1]. At that time, theories of the mutability of species were rumbling through all levels of English society. Such suggestions were in direct confrontation with the teachings of the church and were therefore an intolerable affront to the position of God in the universe. In 1844, *Vestiges of the Natural History of Creation* [2], published anonymously, had scandalized the church and academia, and fanned the flames of unrest in the lower classes. In those times, science and morality were tightly coupled. Many fields of scientific enquiry currently enjoy freedom of thought in an atmosphere that owes much to the revolution wrought by the theory of evolution. It is impossible to fully appreciate the importance and magnitude of these changes without an understanding of the social context in which this theory developed.

Darwin, like others who promoted evolution through natural selection, was by no means infallible. For example, he was unable to demonstrate a mechanism of inheritance in which variation was maintained. His eventual hypothesis on the subject, the theory of *pangenesis* proved incorrect. It was fifty years before the details of hereditary began to fall into place, and another thirty before the "evolutionary synthesis" solidified the link between the theory of evolution and the relatively young field of genetics [3, 4]. Nevertheless, Darwin had identified the central mechanism of evolution: selection in the presence of variation, or "descent with modification" as he called it. In many cases, the specifics of evolution through variation and selection are still being brought to light. However, the basic mechanisms have sufficed to explain, to the satisfaction of many, an incredibly wide range of behaviors and phenomena observed in the natural world.

It is no wonder then that computer scientists have looked to evolution for inspiration. The possibility that a computational system, endowed with simple mechanisms of variation and selection, might be persuaded to exhibit an analog of the power of evolution found in natural systems is an idea with great appeal. This hope has motivated the development of a number of computational systems modeled on the principles of natural selection. Some history of these efforts is described by Goldberg [5] and Fogel [6].

A difficulty with the hope that we might build computational systems based on the principles of natural selection, and put these systems to practical use, is that natural systems are undirected, whereas virtually all our activity is highly directed. We use the computer as a tool for solving specific problems we construct, and we place a lot of emphasis on doing so as quickly as possible and with as little waste as possible. Natural systems have no such aims or constraints. Survival is not directed toward some fixed target. Instead, it is a matter of keeping a step ahead, in any accessible direction. It is a gross generalization, but I believe efforts to date can be grouped into two broad categories:

- Useful systems that are modeled loosely on biological principles. These have been successfully used for tasks such as function optimization, can easily be described in non-biological language, and are often outperformed by algorithms whose motivation is not taken from biology.

- Systems that are biologically more realistic but which have not proved particularly useful. These are more faithful to biological systems and are less directed (or not directed at all). They exhibit complex and interesting behaviors but have yet to be put to much practical use.

Of course, in practice we cannot partition things so neatly. These categories are merely two poles, between which lie many different computational systems. Closer to the first pole are the *evolutionary algorithms*, such as Evolutionary Programming, Genetic Algorithms and Evolution Strategies. Closer to the second pole are systems that may be classified as Artificial Life approaches [7]. Both poles are interesting in their own right, and I have spent time exploring aspects of each. I have nothing against evolutionary algorithms that are used to optimize functions or against artificial life systems that are difficult to put to practical use. Yet it seems to me that we are still rather far from tapping the best of both worlds and the hope of being a part of that research, when and if it happens, is the ultimate motivation for the current work.

This dissertation is primarily concerned with evolutionary algorithms. A non-biological perspective on these algorithms is developed and several consequences of that position are investigated. It is important to remember that this is simply one perspective on these algorithms. They would not exist without the original biological motivation, and that ideal has prompted this investigation. I believe that it is important to maintain the ideal and simultaneously keep careful track of whether real progress has been made. Evolutionary algorithms are examined through the study of their relationship to two things:

1. The biological notion of a fitness landscape.

2. Other search algorithms, particularly various hillclimbers and heuristic state space search algorithms found in artificial intelligence and operations research.

The first of these relationships, the connection between evolutionary algorithms and fitness landscapes, has achieved almost folkloric recognition in the evolutionary computation community. Practically every researcher in that community is familiar with the landscape metaphor and does not express discomfort or concern when the descriptions of the workings of these algorithms employ it extensively, using words such as "peaks," "valleys" and "ridges." Surprisingly however, it is rare, within the field of evolutionary algorithms, to find an actual definition of a fitness landscape. This situation has perhaps developed because everyone grasps the imagery immediately, and the questions that would be asked of a less evocative term are not asked. This dissertation presents a model of landscapes that is general enough to encompass much of what computer scientists would call search, though the model is not restricted to either the field or the viewpoint. Evolutionary algorithms are examined from the perspective the model affords.

The second relationship that is addressed is the connection between evolutionary algorithms and the search algorithms of artificial intelligence and operations research. This relationship is not widely recognized. I believe it is a an important one. Its identification allows discussion of evolutionary algorithms in non-biological terms. This leads to a view of evolutionary algorithms as heuristic state space search algorithms, and this correspondence is used to throw light on the question of what makes a problem hard (or easy) for a genetic algorithm.

The remainder of this chapter provides a brief introduction to evolutionary algorithms and fitness landscapes, the model of computation, a discussion of the effects of choice of language, an argument for simplicity, a brief outline of related work, and a description of the structure of the dissertation.

1.1. Evolutionary Algorithms

There is no strict definition of the class of algorithms known as "evolutionary algorithms." It is not hard to argue that members of the class have little to do with biotic evolution and are not strictly algorithms. Nevertheless, the term exists, and I will attempt to make clear what is usually meant by it. In rough order of appearance, members of the class generally fall into one of the following three categories: Evolutionary Programming [8], Genetic Algorithms [9] and Evolution Strategies [10, 11]. The collective set of approaches has become known as Evolutionary Computation (see [6, 12] for overviews).

Many variants of these methods have emerged. Two of the best known are Messy Genetic Algorithms [13] and Genetic Programming [14, 15], both of which are descendants of genetic algorithms. In describing evolutionary computation, some would fling the net even more widely, and include neural networks, simulated annealing, and many models from Artificial Life. I will not consider these extensions in the following description, but many of the results of this dissertation bear on those fields, and the connections will be detailed in later chapters. The following description of an algorithm could be applied to any of the above evolutionary algorithms or their variants:

> The algorithm maintains a collection of potential solutions to a problem. Some of these possible solutions are used to create new potential solutions through the use of *operators*. Operators act on and produce collections of potential solutions. The potential solutions that an operator acts on are selected on the basis of their quality as solutions to the problem at hand. The algorithm uses this process repeatedly to generate new collections of potential solutions until some stopping criterion is met.

The above description is couched in a deliberately neutral language. In practice, these algorithms are often described in biological terms. The algorithms maintain not a collection of potential solutions, but a *population* of *individuals*. The problem to be solved is given by a *fitness function*. Operators are *genetic operators* such as *mutation*, *recombination* (or *crossover*) and *inversion*. Individuals are chosen to *reproduce* on the basis of *selection* based on their *fitness*. In each case, these algorithms were designed to mimic aspects of evolution through natural selection with the hope that this would lead to general methods that would prove widely useful through similar operation. Although the algorithms represent vast simplifications of the actual evolutionary processes, they have proved widely successful.

1.2. Fitness Landscapes

Using a landscape metaphor to develop insight about the workings of a complex process originates with the work of Sewall Wright in the mid–1920s [16]. By the time Wright first published on the subject, Haldane [17] had begun to think along similar lines. Wright used landscapes to provide easily accessible imagery for his theory of speciation. Since that time, biologists have frequently used the metaphor. Landscapes, according to Eldredge [18] are "By all odds the most important metaphor in macroevolutionary theory of the past fifty years." There are two main interpretations of landscapes in biology. In the first, each member of a population is considered a point on a fitness landscape. This imagery has been used extensively in the development of evolutionary thought. It is the foundation of both Wright's *Shifting Balance* and Simpson's *Quantum Evolution* models of speciation [16, 19]. In the second approach, a point on the landscape is taken to represent the average fitness of an entire population which moves on a surface of mean fitness. Population geneticists have used this viewpoint to prove theories about the effects of selection on average fitness [20, 21].

To biologists, going uphill on a landscape is thought of as increasing fitness. Physicists also find the landscape metaphor useful, but they are usually interested in low energy states of some material (e.g., a spin glass [22, 23]), and so consider lower points on the landscape more desirable. Computer scientists are divided. The conflict is of course a trivial one and it vanishes if one talks exclusively of optimizing on a landscape. I will defer to biology. In this dissertation, algorithms are seeking to find high points on the landscape. Formal discussion of landscapes will begin in Chapter 2.

1.3. Model of Computation

Establishing a formal model of computation is difficult and somewhat arbitrary at the best of times. Even when the computation is being performed in a well-defined manner on a well-defined computational device, there are many subtle issues whose resolution, one way or another, may have decidedly non-subtle effects [24]. It is no surprise then that evolutionary algorithms have not been the subject of wide attention when it comes to models of computation.

When it is discussed, the model of computation that is often used when discussing evolutionary algorithms is some variant of a *black-box* model. See, for example, [5, 25, 26, 27, 28, 29]. In this model, one is presented with a black box whose properties are unknown. The box accepts inputs and each input is rewarded with the value of some mysterious function. The aim of this series of probings is also given, and might be "Find an input which produces a large function value" or possibly "Find an input that produces an output value greater than 100."

Notice that the possible vagueness of the aim of this enterprise may prevent the provider of inputs from knowing when, if ever, it has located an input (or set of inputs) that solve the search problem. To be more specific, I will call the input provider the "searcher" and the task to be addressed the "problem." This is in keeping with the subject matter of the rest of this dissertation: Search algorithms and problems. The black-box model is more commonly phrased in terms of "functions" and "optimization."

Given such a model, the operation of providing an input and receiving an output, often called performing a *function evaluation*, becomes the coin of the realm. Algorithms may be compared on this basis—for example by comparing the number of evaluations required, on average, to achieve a certain performance level, or by computing statistics based on the function values achieved over time [30]. In this dissertation, function evaluations, or, more simply, evaluations, will form the basis for the comparison of algorithm performance. I will adopt a form of black-box model, but before doing so, I wish to argue that without thought, it can be misleading in two important ways:

1. **Representation.** The first problem with the abstract black-box model of computation arises from the fact that we rarely encounter black boxes in the real world. The model assumes that the problem to be solved is already embodied in some mechanical device and that our only task is to play with the input knobs of the box. In reality, we encounter problems in the world outside the computer, and if we intend to attempt to solve them using a computer, we have to get them into the computer. The black-box model is silent on this aspect of computational problem-solving even though this step is an inevitable and important part of the problem solving process. The need to formulate the problem in a fashion that is suitable for computation requires making a decision about how to represent the input to the black box. This is a non-trivial problem. But

the black-box model obscures this requirement, by providing us with a physical object with a conveniently predefined interface. This choice needs to be made every time we address a search problem computationally. The problem of representation has been recognized and debated in the field of evolutionary computation. I am trying to make two points: that the black-box model does not draw attention to this choice and that the choice must always be made.

The choice that is made may have a dramatic impact on how easily the problem will be solved. This adds a very human factor to our otherwise impersonal black-box model. When faced with a search problem such as "Find two integers whose product is 15 and whose sum is 8," few people, presumably, will rush off and construct an elaborate encoding of a pair of integers and write a genetic algorithm to search for the solution. When asked to "Find an arrangement of eight queens on a chessboard such that no queen is attacking any other" it is not unreasonable to use as a representation an eight by eight array of computer words, exactly eight of which are non-zero. Others would immediately see that possible solutions could be represented via a permutation of the integers one to eight, which leads to a far more elegant search algorithm [31]. The same black box could be used to evaluate board configurations for feasibility, but the difference in the choice of representation makes the second approach far more promising than the other. John Von Neumann would perhaps not have seen any need for representation to solve this problem. An extensive discussion of representation and its effects on the difficulty of the "Missionaries and Cannibals" problem is given by Amarel in [32]. Algorithms that operate within the black-box model are often said to be performing *blind* search [5, 27, 28]. The above discussion illustrates that care should be taken with this term since the degree to which the algorithm is fumbling in the dark will very much depend on how much the designer of the algorithm could see.

2. **Evaluation.** The second potential problem with the black-box model concerns the output of the box and is of a similar nature. The severity of this problem depends to some extent on the nature of the original problem statement. In many search problems, no function is given, and again one must make a choice. A good example is the optimization formulation of the NP-Hard Traveling Salesperson Problem (TSP): "Given n cities and their inter-city distances, find a Hamiltonian circuit whose total length is minimal." In this case, we are not told how to recognize a solution and we are certainly *not* given a function that we can interrogate in the black-box fashion.

> In some cases, such as in TSP, a seemingly appropriate function will be so obvious that it will be easy to believe that it was always there. It was not—look at the problem statement. In evolutionary algorithms, TSP has been, at least to my knowledge, universally approached as though a black box was given that took a Hamiltonian circuit as input and returned the total length of that circuit. Again I want to emphasize very strongly that this is a choice (it is often a very good choice). To do this, consider that there is a vastly superior choice: A function which returns the exact number of city exchanges that are required to transform the input into the optimum. Why then do we not use this black box instead of the path length version? After all, the problem could be solved in polynomial time with the simplest form of hillclimbing given such a black box. The reason, of course, is that we cannot *build* such a black box. It should be apparent that the black box is constructed by the problem solver and that we have a choice in how that is done.
>
> A good example of a problem in which we are not given any function is Rubik's Cube. Though this is not a problem (to the best of my knowledge) that has been addressed using evolutionary algorithms, it has received wide attention in the artificial intelligence community. No-one has found a heuristic function whose value can be used successfully by any known search algorithm. Korf [33] presents four heuristic functions for the simplified $2 \times 2 \times 2$ cube, none of which has been used successfully to guide a heuristic search on this simpler problem. To solve Rubik's cube using a non-exhaustive search on a computer, we have to make a choice about how we will assess the worth of the cube configurations we encounter. The reason the puzzle is so hard is that it is not clear how to do this. A similar difficulty was identified for 26 transistor placement heuristics by Shiple et al. [34].

Putting these points together, to use an algorithm that operates via interrogating a black box, we find that we must first construct the black box itself. To make an unqualified claim that an algorithm is worthwhile because of its general-purpose nature and that it can be used in a black-box situation is to ignore two important aspects of problem solving. Instead of stubbing our toe on a black box while strolling casually through a field armed only with a blind algorithm, we build a black box, half-bury it in a field and then, as if by chance, stumble across it and whip out our general-purpose, robust, blind search algorithm and use it to extract the secrets of the black box, after which we proclaim the great and general usefulness of our weapon.

In summary, both these arguments claim that something the black-box model appears to provide actually has to be constructed by the problem solver. These are the form of the input, and the output function. In other words, the entire black box. A naive consideration of the black-box model would take those two as given, whereas I believe that both of them must always be constructed. This leaves us in a curious position. The model of computation we had intended to use has seemingly vanished. There is no black box until the problem solver creates it. The black box model is not independent of the problem to be solved in the way the comparison-based model of traditional analysis of algorithms is [24]. The comparison based model is useful for this reason. It is only fair to compare algorithms based on the number of evaluations they perform if they are using the same input representation and evaluation function.

A graphic illustration of the impact of these choices is provided by comparing exhaustive search on the alternate representations of the eight queens problem mentioned above. One representation has a search space of size 4,426,165,368 while the other is of size 40,320. Even comparing the same algorithm under a different representation leads to a difference in expected performance of five orders of magnitude. If we were to extend such comparisons to different algorithms employing different representations and fitness functions, our results could be even more wildly misleading.

Our model of computation then shall be based on evaluations, and we can use it to compare algorithms, provided that the choices of representation and evaluation function have been made and are the same for the algorithms in question. We will acknowledge the very important human component present in the construction of a black box. The process of choosing representation and choosing an evaluation function requires inferencing and insight into the problem, and may be more important than the eventual choice of algorithm to interpret the black box. This encourages caution when making claims about the virtues of general-purpose, weak methods. These claims tend to ignore the fact that much of problem solving lies in choices that are not made by these weak methods.

1.4. Simplicity

Evolutionary algorithms, though often simple to describe and implement, have proved difficult to understand. Analysis of algorithms (a phrase which, incidentally, has many wonderful anagrams) typically involves the choice of a model of computation, proofs of upper bounds on worst and possibly average-case behavior and proofs of lower bounds on problem difficulty. With luck, better lower bounds will be proved and better algorithms designed until these bounds meet, to within a constant factor, and the problem is considered solved. There are many examples of this process and many of the most basic algorithms in computer science have interesting histories.

The analysis of an evolutionary algorithm is a quite different matter. To begin with, as mentioned above, it is not even clear that they should be regarded as algorithms in the strict sense. Perhaps the biggest reason why analysis cannot proceed as above is that although the algorithm is given in advance, the details of the problem to be solved are not. In the traditional analysis of algorithms, the algorithm being analyzed is an algorithm for a specific problem. In many cases the input to the algorithm is not prespecified, but it is still almost always possible to perform worst-case analysis of the algorithm's behavior. However, worst-case analysis of algorithms that make choices based on pseudo-random numbers is typically not terribly enlightening, and this is certainly also the case with evolutionary algorithms. By considering all possible inputs to an algorithm, average-case behavior can sometimes also be determined (though this is typically a far more demanding task). The same analysis cannot be done for an evolutionary algorithm, because the statement of the algorithm gives no hint as to the identity of a specific problem that is to be solved. This does not preclude analysis completely, it just makes it more difficult and often quite unlike that seen in more traditional analysis.

A second major difficulty that confronts the would-be analyst is the degree to which the evolutionary algorithms are stochastic. In traditional analysis of algorithms, the algorithm always behaves in almost exactly the same way given the same input.[1] In evolutionary algorithms, if this *ever* happened (on a problem of even modest size), it would be considered so improbable that the implementation would immediately be checked for bugs. This massive degree of stochasticity does not lead to algorithms that are straightforward to understand or predict. Intuition is often not a particularly reliable guide to the behavior of a system whose outcome is the result of the composition of thousands or millions of small-scale chance events. A fine illustration of these difficulties is given in work on the so-called Royal Road functions by Mitchell et al. in which a seemingly simple investigation with a predictable outcome produced completely unexpected results [36].

That example had a great effect on me, and marked the beginning of what has become a very pronounced preference towards asking simple questions about these complex algorithms. Simple questions have a reasonable probability of being answerable and have at least the potential for having simple answers. If we ask a simple question of a system and receive a completely unexpected result, it seems logical that we should seriously question our understanding of the system, and that

[1] Quicksort is a well-known example with varying behavior, but the amount of stochasticity in that algorithm is not great enough to make detailed analysis impossible [35].

we should continue to ask simple questions.

This dissertation is a collection of very simple questions and my attempts to answer them. It may appear to some that these simple questions, and the methods that are used to address them, have little to do with evolutionary algorithms. I would argue strongly that this is not the case, and that these questions and the methods used to address them have *everything* to do with the more complicated algorithms. For example, it is important to study the operators of these algorithms in isolation and it is important to study the effect of a population-based search by comparing it with algorithms that have no population.

Amusingly, my taste for asking simple questions is the result of time spent in an environment in which the most complex problems are being addressed. It is not a contradiction to attempt the study of entire complex systems by a form of reductionism that proceeds by asking a lot of simple questions. I regard the development of this taste for simple questions to be (finally) a sign of increased maturity. It is the most important lesson that I will take away from three wonderful years at the Santa Fe Institute.

1.5. Related Work

A discussion of related work would make little sense at this point, and could not be related to the model that will be developed. Related work will be discussed in detail in Chapter 6. That chapter describes how this work relates to (1) Other models of landscapes that have been used in biology, chemistry, computer science and physics, and (2) Other search algorithms, particularly heuristic state space search. A full treatment of these relationships is not possible until the model of this dissertation has been presented and some of its uses have been demonstrated. The body of the dissertation contains many references to and hints at these relationships, though discussion is usually deferred.

1.6. Dissertation Outline

The following chapters can be roughly divided into three sections: formalism (Chapter 2), applications (Chapters 3 and 4 and the second half of Chapter 5) and connection with other search techniques (Chapter 5). Chapter 2 presents the landscape model, and discusses its consequences, advantages, limitations and relevance. Chapter 3 develops the *crossover hillclimbing* algorithm and examines the role of crossover, macromutations and populations in genetic algorithms. Chapter 4 demonstrates one approach to quantifying the behavior of algorithms on landscapes through the use of a *reverse hillclimbing* algorithm that can be used to compute the exact probabilities that a number of hillclimbing algorithms will reach a point on a landscape. Chapter 5 looks at the connection between evolutionary algorithms and heuristic search, and shows the extent to which evolutionary algorithms can be regarded as heuristic state space search algorithms. One aspect of the correspondence is treated in detail and results in a new answer to the question "what makes a problem hard (or easy) for a genetic algorithm?" Chapter 6 examines something of the history of the use of the landscape metaphor in biology and evolutionary algorithms, other related work on landscapes and heuristic search. Finally, Chapter 7 presents a summary of the results of this dissertation.

1.7. Abbreviations

The remainder of this dissertation uses many abbreviations for common phrases and algorithm names. I hope that these will not prove too intimidating to those not already familiar with the various evolutionary algorithms. A complete list of the abbreviations begins on page xxiv.