CHAPTER 6

Related Work

6.1. Introduction

This chapter divides work related to this dissertation into three areas (1) work on landscapes in other fields, (2) work on landscapes in computer science and evolutionary computation, and (3) work related to the connection between landscapes and heuristic search, especially to fitness distance correlation. Several pieces of research discussed below could be placed in more than one of these categories. This is merely one way of roughly grouping related work.

6.2. Related Work on Landscapes

The landscape metaphor originated with the work of Sewall Wright [16]. The idea has received wide attention within biology, but has also been adopted by researchers in other fields. This section provides an outline of the use of landscapes in biology, chemistry, computer science and physics.

6.2.1. Landscapes in Biology

Provine [87] has done an excellent job of summarizing the uses (and abuses) of Wright's "surfaces of selective value." This section merely touches on some of the points that are so well made by Provine. Wright's original diagrams were published in 1932 [16] and were intended to make it simpler to understand his earlier and far more mathematically sophisticated paper on population genetics [161]. The diagrams used by Wright became a popular vehicle for explanations of various phenomena in biology and have become known as "fitness landscapes." Unfortunately, they were widely used in at least three ways:

- In Wright's 1932 paper, each axis of his landscape corresponded to a gene combination. An additional dimension was added for fitness. Thus a point on a landscape corresponded to an individual gene combination and there was one axis for every possible combination. Because each "axis" corresponds to the genetic makeup of an individual, it is not clear how to interpret them as axes. The axes have no units and no gradations. Here is Provine's summary of Wright's original diagrams:

  > "Thus the famous diagrams of Wright's 1932 paper, certainly the most popular of all graphic representations of evolutionary biology in the twentieth century, are meaningless in any precise sense."
  >
  > Provine [87, page 310].

  In defense, Wright [37] acknowledges that his two-dimensional diagrams are "useless for mathematical purposes," but argues that they were never intended for such use. Rather, the diagrams were intended as a convenient pictorial representation of a process taking place in a high-dimensional space.

  Although these first published landscapes make little sense mathematically, the idea of the landscape above fits into the landscape model of this dissertation. Each genetic combination is represented as a vertex in a landscape graph and each is assigned a fitness that is used to label the vertex. The advantage of the new model is that it requires no notion of dimensionality, and can therefore be used naturally to produce a landscape graph that is perfectly well-defined.

- In a second interpretation of his landscapes, Wright regarded each axis as corresponding to the frequency of a give gene. Thus the landscape had as many axis as there were genes, and each axis went from zero to one according to the frequency of the gene in a population. A point in this space therefore corresponds to the gene frequencies of a population of organisms and fitness, the extra dimension, is given by the mean fitness of this population. Many populations might have the same gene frequencies however, and this leads to difficulties with this interpretation. This interpretation predates the one published in 1932. Wright tended to use this version mathematically, while continuing to publish the 1932 diagrams.

  The landscape model of this dissertation can be used to describe this form of Wright's landscapes if one views the process that creates one population from another as an operator that produces edges in a graph whose vertices correspond to entire populations. More in line with Wright's formulation would be a landscape in which a vertex corresponds to the infinite set of populations that exhibit a given distribution of gene frequencies. As Provine observes, this greatly complicates matters when one tries to assign a "fitness" to a vertex, [87].

- Simpson [19] was responsible for the introduction of a third interpretation in which axes measured structural (i.e., phenotypic) variation in individuals. In this version it appears that Simpson intended points in the space to correspond to individuals, with the extra dimension given by individual fitness. This landscape can be viewed as a graph in the model of this dissertation in an analogous fashion to that in which Wright's first version is described above.

These conflicting interpretations of fitness landscapes suggest that the metaphor was an attractive one to many. The lack of a rigorous definition did not prevent use of the landscape metaphor from becoming widespread. Actually, lack of definition may have encouraged this. There is nothing inherently wrong with this situation if landscapes are used to convey a vague picture of some process. When attempts are made to use landscapes in a rigorous fashion, it becomes more important to know what is being referred to. A similar situation exists in evolutionary computation, where the word landscape tends to be used in a rather cavalier fashion. Landscapes are frequently heard of but infrequently defined. In some instances this is very useful. In other cases, landscapes are used as the foundation of seemingly plausible arguments that, when closely examined, can be extremely difficult to make any sense of at all.

6.2.2. Landscapes in Physics and Chemistry

Two fields in which landscapes are used that do not suffer from a lack of formal definition are physics and chemistry. Formal definitions can be found in the work of Weinberger [63, 64] and in a number of papers by Fontana, Huynen, Schuster, Stadler and others in theoretical biology and chemistry (see [162, 163], and references below, for examples). Weinberger's landscapes are described in the following section. Both these models were formulated to study specific systems; the dynamics of spin glasses and the folding and evolution of RNA sequences and secondary structure. They are clearly defined and are useful for the study of these systems. However, not surprisingly, they are not directly applicable to evolutionary algorithms. There are three primary reasons for this:

- These models are only relevant to systems in which an operator acts on a single "individual" to produce another "individual." That is, one RNA sequence is converted to another, or one Hamiltonian circuit in a graph is converted into another.

- The systems under consideration all involve a single operator. For example, an operator which changes a spin in a spin glass or a point mutation operator which changes a nucleotide in an RNA molecule.

- All possible outcomes from an application of an operator have equal probability of occurrence.

These differences make the application of these landscape models to evolutionary algorithms problematic. In evolutionary algorithms, none of the above are true. These algorithms have operators that act on and produce multiple individuals, they employ multiple operators, and the possible results of operator application are not equally probable. These possibilities are incorporated in the landscapes model of this dissertation.

6.2.3. Landscapes in Computer Science

Weinberger's work has had a significant influence on work done in computer science [63, 64, 65, 164]. Weinberger provides a precise definition of a fitness landscape. His landscape is also a graph (though finite), whose vertices are labeled using some real-valued function. He introduces the autocorrelation function and shows its use as a measure of landscape fitness correlation or "ruggedness." This measure and others derived from this work have been relatively widely used [60, 61, 62, 70, 117, 118, 165]. The model presented in this dissertation generalizes Weinberger's model. It allows operators that act on and produce multisets of points in $\mathcal{R}$, allows the graph to be infinite and/or unconnected, and assigns probabilities to edges which is important when considering mutation or crossover. Figure 1 on page 31 illustrates this common situation with crossover. Weinberger considered operators that would assign an equal probability to each outgoing edge from a vertex, and so had no need for probabilities.

Manderick, De Weger and Spiessens [60] used Weinberger's landscape and autocorrelation function to examine the relationship between the statistical structure of landscapes and the performance of genetic algorithms on the same landscapes. This important paper illustrates how successfully statistics concerning landscape structure can predict algorithm performance. Manderick et al. viewed a landscape as something defined by an operator that acted on and produced a single point of $\mathcal{R}$. Weinberger's autocorrelation function can be calculated for these simplest of operators. They recognized crossover as an operator without making the generalization of the model of this dissertation. To deal with this, they defined a measure of operator correlation which they applied successfully to one-point crossover on NK landscapes and to four crossover operators for TSP. This statistic is calculated by repeated use of the operator from randomly chosen starting points. Thus, they were actually computing statistical measures about the crossover landscape without recognizing it as a landscape. Had they done so, they might have used the autocorrelation function, since they dealt with the walkable landscapes generated by crossover operators that take two parents and produce two children.

Taking the work of Manderick et al. as a starting point, Mathias and Whitley [61] examined other crossover operators for TSP and Dzubera and Whitley develop measures of "part" and "partial" correlation which they also apply to TSP. Much of the work mentioned above in theoretical chemistry has also been influenced by Weinberger's work (see, for example, [62, 166, 167, 168]). An overview of uses of the landscape perspective is far beyond the scope of this section. These works are mentioned to show the extent of the influence of Weinberger's model and definition of the autocorrelation function and correlation length.

The work of Nix and Vose and others on Markov chain analysis of GAs [52, 169, 170, 171, 172, 115] can also be looked at from a landscape perspective. As mentioned in §2.3(23), the choice of what one regards as an operator is a matter of perspective. Different choices will impact the ease with which we can study different aspects of search algorithms. By viewing an entire generation of a GA as an operator that converts one population to another, the GA can be imagined as taking a walk on a landscape graph whose vertices correspond to populations. This operator, like any other, can also be viewed as defining a transition matrix for a Markov chain. Naturally, for a process as complex as an entire GA generation, calculating the transition probabilities will be quite involved, but this is what Nix and Vose have done. Because this operator is used repeatedly (unlike simpler operators such as crossover and mutation which are used in series), the expected behavior of a GA can be determined via Markovian analysis. Due to the exponential growth in the number of possible states of this Markov process, such analysis is typically restricted to small populations consisting of short individuals. Nevertheless, it offers exact results and insights that are otherwise not available. Markov chain-like analysis of individual operators has been carried out by Goldberg and Segrest [173], Mahfoud [174], Horn [175] and Whitley and Yoo [176].

Recently, Culberson [29] independently conceived of a crossover landscape. His structure, which he calls a *search space structure*, is also a graph and the vertices correspond to a population of points from $\{0, 1\}^n$. He examines crossovers between two complementary strings which creates a graph that corresponds to the largest connected component of the crossover landscape generated by $\chi_1^{2\to 2}$ (two parents, two offspring, one crossover point) in the current model. That component (like all others) is a hypercube. It contains vertices that correspond to all possible pairs of binary strings of the form $(a, \overline{a})$. Culberson shows that the structure of the component is isomorphic to the hypercube generated by the bit-flipping operator for strings of length $n - 1$. He shows the existence of $\chi_1^{2\to 2}$-local-minima in this structure and demonstrates how to transform a problem that appears hard for one operator into a problem that appears hard for the other. This provides further evidence of the importance of structure for search and of how that structure is induced by the choice of operator. Wagner is also investigating isomorphisms between crossover and mutation landscapes [177].

There is other related work whose connections to the landscape model have not yet been well explored. This includes work done by Whitley and by Vose on mixing matrices [178, 179], work by Altenberg [180], whose transmission functions for crossover operators in $\phi_{2\to 1}$ can be drawn as landscape graphs similar to that shown in Figure 11 on page 54, and Helman's work on a general algebra for search problems [140].

6.3. Work Related to the Heuristic Search Connection

Tackett [119, 120] proposes that there is a connection between Genetic Programming (and thus GAs) and heuristic search. In particular, he outlines a correspondence between GP and beam search [181]. The correspondence is based on the fact that the population in a GA can be viewed as akin to the priority queue of beam search. Population members in a GA are "expanded" (chosen to produce offspring) in a fashion prioritized by fitness. In beam search, nodes (individuals) are expanded strictly according to position in the priority queue. This correspondence fits perfectly into the correspondence between evolutionary algorithms and heuristic search developed in Chapter 5 using the language changes summarized in Table 31 on page 125. Tackett [119] also makes the point that the uniform use of biological language to describe GAs tends to obscure the connection with heuristic search.

6.4. Work Related to FDC

There are four pieces of work (that I am aware of) which present diagrams similar to those used to compute fitness distance correlation. In each case, the diagrams are closely related to the scatter plots used for FDC, but the uses to which they are put (if any) and the research they motivate are all quite different.

1. Kauffman ([117, page 564] and [118, page 61]) shows scatter plots of the fitness of local minima versus their distance from the best local minimum found, for several NK landscapes. A number of local minima are discovered through hillclimbing, and each is represented by a single point in the plot. These diagrams demonstrate the existence of what Kauffman calls a "Massif Central" in NK landscapes with small K. By this it is meant that the highest peaks in the landscape are near each other and this region occupies a very small region of the entire search space. These scatter plots differ from the plots used in FDC only in the fact that each point in the plots corresponds to a peak on the landscape, not to a randomly chosen vertex in the landscape. Kauffman does not compute correlation or other statistics about the relationship between distance from (apparent) global optimum and fitness of local optima.

2. Boese, Kahng and Muddu [116] produce similar scatter plots, but on a variety of different problems, and use them to motivate the construction of "multi-start" hillclimbers that take advantage of the "big valley" structure they observe. Their "big valley" is Kauffman's "Massif Central" inverted (Boese et al. study minimization problems). They observe that the highest peaks (found by various hillclimbers based on different heuristics) are tightly clustered. They use this knowledge to construct an "adaptive multi-start" hillclimber that takes uphill steps in the same way the BH algorithm of §3.6(55) does. The hillclimber uses the 2-*opt* operator introduced by Lin and Kernighan [182]. The algorithm assumes that the problem at hand exhibits the big valley structure, and they show how the hillclimber compares very favorably with other hillclimbers that do not take advantage of this global structure. The hillclimber uses information about high fitness peaks located earlier in the search to select new starting vertices on subsequent climbs. This technique is applied to 100- and 500-city randomly generated symmetric traveling salesperson problems and to 100- and 150-vertex graph bisection problems. Boese [183] extends this work to consider the 532-city traveling salesperson problem of Padberg and Rinaldi [184] and

mentions that similar plots have been obtained on circuit partitioning, graph partitioning, number partitioning, satisfiability, and job shop scheduling.

In addition to scatter plots showing fitness of local optima against distance to global optimum, Boese et al. produced plots of fitness of local optima versus mean distance to other local optima. On the problems they have studied, there has (apparently) been a single global optimum, rather than many, as with some problems studied with FDC (which uses distance to the nearest global optimum). Boese [183] produced similar plots on the 532-city problem using local optima located via several hillclimbers, each using a different "heuristic" ("operator" in the language of this dissertation). The results for these different hillclimbers allow Boese to compute the size of the "big valley" for each operator. The correlation between fitness of local optima and distance to global optimum was also computed and used to examine the relative strengths of the relationship for each of the operators.

Boese et al. use a distance metric that is not the same as the distance defined by the operators they employ. There is no known polynomial time algorithm for computing the minimum number of 2-*opt* moves between two circuits in a graph. Instead, they use "bond" distance which is the number of vertices in the graph minus the number of edges the two circuits have in common. Some analysis shows that this distance is always within a factor of two of the 2-*opt* distance between the circuits. This is similar to the use of Hamming distance instead of "mutation distance" or "crossover distance" in the calculation of FDC.

Because the scatter plots of Boese et al. examine local optima, the relationship that will be observed between fitness and distance will depend on the difficulty of the problem and on the quality of the local optima that are examined. If a good hillclimber is used to locate local optima on an easy problem, few local optima may be found. In the worst case, on a non-pathological (in the style of Horn et al. [86]) unimodal problem, only one peak will be located, and it will be the global optimum. Boese et al. and Hagen and Kahng [185] have observed instances where the relationship between fitness and distance deteriorates and this appears dependent not only on the problem at hand, but on the quality of the heuristic used to locate local optima. FDC does not run into problems in these situations because it considers randomly chosen vertices of the landscape—there is no requirement that they be local optima. Thus FDC is useful for classifying easy problems as well as difficult ones. This is of little consequence to Boese et al., who are motivated to take advantage of the big valley structure in problems that contain many local optima. Naturally, the pervasiveness of the "big valley" structure across problem types and instances will determine how useful the adaptive multi-start algorithm is. Judging from these initial investigations, the class of problems possessing this structural property may quite large. If so, algorithms designed to exploit this structure, such as adaptive multi-start, will be hard to beat.

3. Ruml, Ngo, Marks and Shieber have also produced a scatter plot of fitness against distance from a known optimum [186]. They examined an instance of the number partitioning problem (of size 100) and compared points in several representation spaces to the optimum obtained by Karmarkar and Karp [187] (KK-optimum). Ruml et al. generate points by taking directed walks away from the global optimum, at each step increasing the distance from the KK-optimum by one. The points found on several such walks are then plotted as fitness (normalized by the fitness of the KK-optimum) versus distance from the KK-optimum. The resulting plots exhibit a "bumpy funnel" in which there is a strong correspondence between solution quality and distance from the KK-optimum. As with the observations of Boese et al. [116, 183], the high quality solutions are tightly clustered around the best-known optimum and Ruml et al. suggest that search algorithms that take advantage of this type of structure (which they term "gravitation") will prove competitive. It is clear from these results that the problem structure around the KK-optimum is qualitatively like that around the optima studied by Boese et al.. Unlike Boese's results with the 532-city TSP problem, we cannot be so confident that the KK-optimum is the global optimum as Karmarkar and Karp's algorithm exhibits a strong bias towards equal-sized partitions. The relationship between fitness and distance is not quantified in any way—the scatter plots are used to give pictorial evidence of the structure of the problem and "gravitation."

4. As mentioned in §1.3(8), Korf presented several heuristics for the $2 \times 2 \times 2$ version of Rubik's cube [33]. Korf's graphs have heuristic value on the X axis and the mean distance of all cube configurations with a given heuristic value on the Y axis. At first glance, these graphs would seem to have little to do with the FDC scatter plots of Chapter 5, but this is not the case. If the axes are swapped and each point is replaced by the actual set of points that it represents, the result is a scatter plot. As Korf points out, there will be virtually no correlation between fitness and distance in any of these plots. This replacement and inversion of axes has not been done, but Korf's diagrams make it apparent

that if correlation is not zero, it can only be slightly negative (this is undesirable in Korf's heuristic functions, which must be maximized).

Finally, there is a relationship between FDC and Weinberger's correlation length [65]. Correlation length is a measure of correlation between fitness values, but the autocorrelation function, from which correlation length is calculated, also takes distance into account. The autocorrelation function provides the correlation between fitnesses at all distances. The connection between this measure and FDC is the subject of current investigation. In particular, if FDC proves useful for prediction using the method suggested in §5.7.2(160), it will likely be prone to "error" in the way that correlation length is when the fitness landscape is not isotropic. When FDC is computed from the point of view of a randomly chosen vertex (or peak), it, like correlation length, will indicate that fully deceptive problems (for example) are easy. In practice there may be nothing wrong with this, since if we ever encounter a reasonably large fully deceptive problem, it would be a simple problem, as far as we would ever be able to tell. The global maximum may as well not exist. A measure of problem difficulty that accords well with what we actually experience is likely to be far more useful than some measure which insists that such a problem is actually very hard.

CHAPTER 7

Conclusions

In the introduction to this work, I wrote that the dissertation was a collection of very simple questions and my attempts to answer them. It seems appropriate then that a concluding chapter should present these questions and summarize (1) the answers I have found, (2) the increase in understanding that the answers bring, and (3) the practical benefits that result from the understanding.

What is a landscape?

This is the most fundamental question of this dissertation. In the model of this dissertation, a landscape is a labeled, directed, graph. The vertices of the graph correspond to multisets of individuals and the edges correspond to the actions of an operator. If an operator acts on an individual A, producing individual B with probability p, then the landscape graph contains an edge from the vertex representing A to the vertex representing B and that edge is labeled with p. It is important to allow a vertex to correspond to a multiset of individuals, not just a single one. By so doing, landscape graphs are well defined for operators that act on and produce multiple individuals. The vertices of a landscape graph are labeled with a value that can be thought of as a height, and this gives rise to the landscape image. When vertices correspond to single individuals, they are typically labeled with values from a fitness function. If not, the label is some function of the fitnesses of the individuals represented by the vertex. This, in essence, is the landscape model of this dissertation.

When evolutionary algorithms are viewed from the perspective of this model, there is an important consequence. Because these algorithms are usually viewed as making use of several operators, they are not operating on a single landscape, but on many. As a result, there are mutation landscapes, crossover landscapes, selection landscapes—every operator creates a landscape graph. Every time an algorithm uses a particular operator, it can be seen as traversing an edge, or taking a "step," on the landscape defined by the operator in question. I call this the "One Operator, One Landscape" consequence of the current model. If we intend to study and talk about landscapes, my model insists that we pay attention to the "one operator, one landscape" consequence. Naturally, we can study the results of using several operators in concert, but we can ask simpler questions—questions about the individual landscapes.

There are other consequences of the model, and these are described in §2.7(36). Two of these are the fact that landscapes may not be connected, and that they may not be *walkable*.

The model of landscapes presented in this dissertation was designed to be useful for thinking about evolutionary algorithms. Other models, designed for different purposes, have limitations that make them inappropriate for this purpose. The difficulties are that these other models do not (1) allow for algorithms that employ multiple operators, (2) account for operator transition probabilities, or (3) admit operators that act on or produce multiple individuals.

What Makes a Search Algorithm?

The landscape model is a very general one, and its usefulness is not restricted to evolutionary algorithms. The model is presented in terms of search, which leads one to ask what are the components of a search algorithm. If we suppose that the task of a search algorithm is to find some object or objects, we can ask what kinds of objects are examined by the algorithm and what its methods are for generating new objects when looking for the solution. The landscape model regards the methods used for creating new objects to examine as operators. Each operator creates the edges in a landscape graph, the vertices of which correspond to the collections (or multisets) of objects that the operator acts on and produces.

But a search algorithm is more than a collection of landscape graphs. The algorithm must explore these graphs in search of a solution object. The landscape model tells us nothing about this aspect of search. From the point of view of the model, search algorithms can be divided into landscapes and everything else. I call the "everything else" the *navigation strategy* of the search algorithm. The navigation strategy determines how and in what order the graphs are explored, from where exploration should commence, how it should be altered as the search proceeds, and when it should be halted. Thus, search algorithms may be thought of as composed of structure(s) and strategy.

It is reassuring to note that this perspective, though arrived at through thinking about evolutionary algorithms and landscapes, is nothing more than the perspective on search that has long been held in the Artificial Intelligence community. For example, production systems have been described as consisting of a "database," "production rules," and a "control strategy." These correspond fairly well to our landscape view of search with its vertices, edges (operators), and navigation strategy. Establishing links with the Artificial Intelligence search community is a good reality check, and, as will be seen, these links are more than superficial.

Can the Pieces be Reassembled?

If search algorithms can be decomposed into landscape structures and strategies, can we take several search algorithms apart and reassemble them into new algorithms? The answer is that we can do this and that it is easy. The reason is that landscapes are graphs and navigation strategies are designed to search graphs. So long as the landscape graphs of one algorithm satisfy any assumptions made by the navigation strategy of another algorithm (e.g., that the graph is a tree or that it is connected), there seems to be no logical reason to prevent the combination of pieces of search algorithms in any way we choose. With nothing more than the perspective offered by the landscape model, a vast number of possible new algorithms suggest themselves. Hybrid search algorithms are nothing new, but the landscape model encourages this and gives a formal framework for doing it. The field of possibilities can be explored systematically, rather than through investigations that follow occasional creative leaps.

Though this perspective might be obvious to those who study search in other contexts, it suggests novel ideas for hybrid algorithms whose pieces are drawn from evolutionary algorithms. The model seamlessly accommodates operators that act on multiple individuals, and so there is no difficulty exploring the graphs that correspond to these operators. To anyone familiar with these algorithms, and with the important unanswered questions about them, an attractive first choice of targets is an algorithm that explores the crossover landscape of a genetic algorithm. The simplest (interesting) navigation strategy is some form of hillclimbing, and so is born the *crossover hillclimber*.

The crossover hillclimber uses a simple hillclimbing strategy to explore the landscape graph of a crossover operator (§3.7(57)). On virtually all of the problems studied in Chapter 3, the algorithm performs better than either the bit-flipping hillclimbing algorithm from which the navigation strategy was taken, or the genetic algorithm from which the landscape was taken. The division into structure and strategy can be made, the pieces can be combined in novel ways, and the resulting algorithms can be practical. The success of the crossover hillclimber leads to another simple question.

Is Crossover Useful?

This question has no simple answer because it is too general. It is tautological to answer that crossover is useful if it manages to orchestrate the exchange of building blocks between individuals. The question we should be asking is: "Given a particular set of choices about algorithm, representation and fitness function etc., does the use of crossover produce benefits that other, simpler, operators cannot produce?" If so, there is a good argument for the use of crossover in that situation. A more complex subsequent question could ask for the characteristics that these situations have in common. In other words, as Eshelman and Schaffer asked, what is crossover's niche? [89].

From a cursory inspection of the crossover hillclimbing algorithm, it appears that the algorithm has isolated crossover from a genetic algorithm and harnessed its power by searching using a more exploitative navigation strategy. New individuals are either created completely at random or via the use of crossover. As it is easily shown that the randomly created individuals are of predictably low quality, the credit for success of the algorithm must go to crossover. This reasoning is correct, but it is far from being the whole story.

Closer examination of the crossover hillclimbing algorithm reveals that its performance correlates extremely well with the number of crossovers it performs in which one individual is randomly created (§3.10(72)). But crossovers involving one random individual are nothing more than macromutations of the non-random individual, where the mutations are distributed in the fashion of the crossover operator in question. The conclusion that crossover hillclimbing was successful because it was making macromutations was confirmed when an extreme version of the algorithm, in which every crossover involved one random individual, proved better than all previous versions of the algorithm. This leads to a distinction between the *idea* and the *mechanics* of crossover. The idea is what crossover is trying to achieve (the recombination of above average genetic material from parents of above average fitness) and the mechanics is the method by which this is attempted. All crossover operators share the same basic idea, but their mechanics vary considerably. The crossover hillclimbing algorithm is a clear demonstration that crossover can be used effectively for search even when it is only the mechanics that are doing the work.

Can the Usefulness of Crossover be Tested?

Crossover can be useful in two ways, one of which is purely macromutational. To decide whether crossover is more useful than other, simpler, operators therefore requires answering whether the idea of crossover is producing gains over what could be obtained through macromutation alone. The standard test for determining if crossover is useful cannot answer this question. The standard test involves comparing a genetic algorithm with crossover to one without crossover. This is a comparison between an algorithm with both the idea of crossover and the mechanics that attempt to implement it, and an algorithm with neither of these. If the genetic algorithm with crossover does better than the one with no crossover, the conclusion that crossover is therefore useful (i.e., that it is doing more to aid the search than a simple macromutational operator would) is not justified. Enter the headless chicken.

The headless chicken test, in which a genetic algorithm with crossover is compared to one with *random crossover* is more informative (§3.12(77)). In random crossover, two random individuals are generated and used in normal crossovers with the parents. One offspring from each of these crossovers is passed back to the genetic algorithm. There is no communication between members of the population. The random crossover operator throws out the idea of crossover while keeping the mechanics. If the genetic algorithm with crossover does not outperform the genetic algorithm with random crossover, then by our definition of usefulness, crossover is not useful. Crossover is not producing gains that could not be obtained with a simpler operator (in this case macromutation). If the genetic algorithm with crossover fails the test (i.e., it does not outperform the genetic algorithm with random crossover), we cannot conclude that the genetic algorithm with crossover is *not* combining building blocks. The test simply shows that the rate at which this is being done (if at all) is not sufficient to produce gains that are greater than those that can be obtained through simple macromutation.

When the genetic algorithm with crossover does not outperform the genetic algorithm with random crossover on a particular combination of representation, fitness function etc., it is difficult to argue that the genetic algorithm *in that configuration* is making effective use of the population. If the genetic algorithm with crossover is worse than the genetic algorithm with random crossover, the population may actually be *hindering* the search. Not only are building blocks not being combined at any significant rate, but the possibility of advancing the search via macromutations is reduced because a population is used. The random crossover algorithm has the benefit of being able to conduct wider exploration. It is not constrained to drawing partners for crossover from a partially converged population whose makeup varies relatively slowly over time.

If crossover is not useful, the usefulness of the population and therefore of the genetic algorithm is called into question. If the genetic algorithm does not pass the headless chicken test, it is either time to look for alternate representations, fitness functions and operators etc. for the genetic algorithm, or, it is time to look for a new algorithm.

What is a Basin Of Attraction?

One of the purposes of using the landscape metaphor is to discuss properties of landscapes. To some extent, landscapes can be studied independently of the search algorithms that will be used to explore them. While it is true that all the components of a landscape (vertices, edges, vertex labels and transition probabilities) are determined by choices that are part of a search algorithm, a particular set of choices may be made by many algorithms. Properties of a particular landscape will be relevant to the entire family of algorithms that make the set of choices that determine the landscape. Algorithms which make very similar choices will perhaps be operating on similar landscapes, and these landscapes can be compared.

A feature of landscapes that is often discussed are the basins of attraction of peaks. Given a landscape, the basin of attraction of a vertex v in landscape graph is the set of vertices that are connected to v by a path (directed towards v). As edges correspond to applications of an operator, the basin of attraction is the set of vertices from which a random walk, using the operator, could possibly reach v.

Natural questions about the basin of attraction $B_\phi(v)$ of a vertex v include the size of $B_\phi(v)$, the identity of the vertices in $B_\phi(v)$, the number of vertices $B_\phi(v)$ has in common with other basins, the probability that vertices in $B_\phi(v)$ will reach v, and the overall probability that v will be discovered via a random walk, using ϕ, from a randomly chosen starting vertex in the landscape. Many of these questions can be answered by the *reverse hillclimbing* algorithm developed in Chapter 4. Given a set of peaks on a landscape, reverse hillclimbing can discover their basins of attraction for a number of hillclimbers and use these results to indicate which of these has the highest probability of locating a peak in the desired set. An exact answer is provided (assuming the number of vertices in the basins are not so great as to exhaust available computational resources). Using methods based on (non-exhaustive) sampling to obtain this information is computationally prohibitive and the answers are necessarily inexact. Chapter 4 and the results in [110] show how the reverse hillclimbing method can be used to reach rapid and accurate conclusions about hillclimbers based on data about basins of attraction.

What Makes Search Hard?

This is the most important question of this dissertation. This question is extremely hard to answer in general, and it is made harder by the fact that hardness is in the eye of the beholder. If one algorithm finds a problem hard but another does not, is the problem hard? If one algorithm never solves a problem with a given level of computational resources, but always solves the problem when the resources available are doubled, is the problem hard? The theory of NP-completeness [141, 188, 189] has produced many beautiful results concerned with problem difficulty, but the fact that a problem *type* is NP-complete does not necessarily tell us anything about a given *instance* of that problem [190]. The Halting Problem is formally undecidable, but faced with a particular Turing machine, it may be trivial to determine whether it halts. Additionally, we may not be concerned with finding the optimal solution to a problem. We may be satisfied by solutions that are very far from optimal. How hard are these problems?

If, instead, we ask "What properties of a landscape make it hard to search?" we can make significant progress. By ignoring algorithms (apart from the choices they make that determine landscapes), the problem becomes at once easier and harder. We may decide that a certain landscape feature, for instance "ruggedness," makes search difficult. If, in some quantitative sense, a landscape can be classed as "rugged" (and therefore difficult), does that mean that all search algorithms that operate on that landscape will have difficulty? No, because search algorithms are more than landscapes; they also have navigation strategies. What is difficult for one navigation strategy may be simple for another. On the other hand, our task is simpler as we are ignoring navigation strategies and it should be possible to develop statistical measurements of landscape graphs that might correlate well with observed algorithm difficulty.

My answer to this question can be arrived at most simply by considering what properties a landscape might have that would make search *easy*. The vertices of any connected graph can be labeled with the length of the shortest path to a designated vertex (whose label is therefore zero). If the designated vertex corresponds to the optimal solution to a search problem the landscape becomes unimodal. If a function of this nature could be found for any instance of a problem, it could be used to solve SATISFIABILITY in polynomial time. We obviously cannot hope to find such a (fitness) function, but it might be reasonable to hope that the more closely our fitness function and landscape approximate this ideal, the easier search will be on the landscape.

A measure of this, Fitness Distance Correlation (FDC), is defined in Chapter 5. The application of FDC to approximately 20 problems from the literature on genetic algorithms shows it to be a very reliable measure of difficulty for the genetic algorithm. Correlation is only one summary of the relationship between fitness and distance from the goal of the search, and in some cases it is too simplistic. Scatter plots of sampled vertices on a landscape often reveal structure not detected by correlation. These plots are one way of looking for structure in a landscape graph and they are often surprisingly revealing. Apart from correctly classifying many problems correctly, including ones that gave surprising results when first studied. FDC also predicted that the question of whether or not Gray coding was beneficial to a genetic algorithm was dependent on the number of bits used in the encoding. The accuracy of this prediction was later confirmed experimentally.

FDC captures a property of a landscape graph. It has nothing to do with any search algorithm beyond the choices that determine the landscape. As a result, FDC can indicate that a problem should be easy, but an algorithm, because of its navigation strategy, may find the problem hard. The indications of FDC about a problem can be interpreted as a rough measure of how hard a problem should be. If FDC says a problem is easy but an algorithm does not find it so, this may be taken as a sign that this choice of algorithm (or algorithm parameters) is unusually bad for this problem. As an indicator of genetic algorithm performance, FDC has proved very reliable.

Are Evolutionary and Other Search Algorithms Related?

The development of the landscape model was driven by the characteristics of evolutionary search algorithms. Because it is quite general, it is worth taking the trouble to examine how search on a landscape resembles ideas of search in other domains. The division of search algorithms into navigation strategy and landscape graphs was shown above to correspond to the Artificial Intelligence community's view of search. A look at the details of heuristic search in a state space, particularly a look at earlier work [43, 142, 191], reveals that the correspondence between this form of search and search in the landscape model is rather strong.

The connection is so strong that much of what needs to be done to move between the fields can be accomplished by a simple change in the language used to describe algorithms. Substituting "state space" for "landscape," "potential solution" for "individual," "control strategy" for "navigation strategy," "heuristic function" for "fitness function," "goal" for "global optimum," and "OPEN list" for "population," much of what needs to be changed is achieved. Naturally, there are differences in these algorithms, but the similarities are very strong. Areas of difference do not so much weaken the argument of a correspondence as they indicate areas of potentially fruitful interaction between the fields.

As an example of this, consider heuristic functions and fitness functions. Both are used to label graphs in which search takes place. One big difference is the amount of research that has gone into the theory of these functions. Heuristic functions have been studied for almost 30 years [142] and there are many results regarding their properties and desirable characteristics [31]. In the world of evolutionary algorithms, fitness functions are relatively unexamined. The choice of fitness function is rarely even acknowledged as a choice. Can we transfer some of the knowledge from the theory of heuristic functions to help us with a theory of fitness functions?

The answer is Yes, and the evidence is the FDC measure. The idealized fitness function described in the previous section is not really a fitness function. A fitness function typically assigns a fixed fitness to an individual, regardless of where that individual is located. The ideal function above assigns "fitnesses" according to distance from the global optimum. But this is *exactly* what Artificial Intelligence has been explicitly seeking in heuristic functions since they were first introduced [142]. To question how well a fitness function approximates the ideal is to question how close they are to the Artificial Intelligence ideal. What we would like is not a fitness function but an heuristic function. The FDC measure arose from considering fitness functions as heuristic functions and asking what sort of a job they did. Questions about admissibility, monotonicity and informedness, from the theory of heuristic functions, can be put to fitness functions. In particular, theory of non-admissible heuristics may inform the study of fitness functions in evolutionary algorithms.