CHAPTER 2

A Model of Landscapes

This chapter presents the landscape model that will be used as a framework for the remainder of the dissertation. It is assumed that one's aim is to address a search problem. It is possible to adopt more neutral language, to the point that the model seems completely abstract. However, adopting the language of search does not restrict applicability too severely. One argument for neutrality is that since the model is applicable to situations that some people would not consider search, it should be described in a general way that will permit its easy application to these fields. A counter argument holds that most of what we do, especially with computers, can be phrased in terms of search. If the landscape model presented here is useful in some field of endeavor, then it is probably possible to describe that endeavor as a search. As will be seen, choosing to view a wide range of situations from a search perspective is not without precedent.

2.1. Motivation

Given the aim of this chapter, to present a model of landscapes, there are many preliminary questions that should be addressed before the model is presented. Three of these are particularly important.

- **What is a landscape?**
  A landscape is one way to view some aspects of a complex process. As such, it is a tool that we hope to use to increase understanding of the process, to suggest explanations and ideas, and to provide an intuitive feeling for those aspects of the process. This description is vague because landscapes may be used in a wide variety of situations. Wright employed the metaphor to provide a simplified and intuitive picture of his work on the mathematics of gene flow in a population [37]. We will use landscapes to produce a simplified view of some

aspects of search. The metaphor has been used in this way for some time to describe evolutionary algorithms, though rather informally. As a result, I think the current use of the metaphor does not always produce beneficial results. Support for this opinion will be presented shortly.

It is important to understand what I believe a landscape is and how that will affect the presentation of the model in this chapter. In particular, a landscape is neither a search algorithm nor a search problem. As a result, to formally describe a landscape, it is not necessary to formally describe all aspects of all search algorithms or all aspects of all search problems. For this reason, we will examine search algorithms and problems of a certain kinds and will omit many details of algorithms and problems. The landscape model will be widely applicable to search, but it is far from a complete picture of search.

A landscape is a structure that results from some of the choices (by no means all) that are made when we use a computer to search. It presents only a simplified view of aspects of the search process. A landscape is a tool and it is a convenience. The model of landscapes presented in this chapter aims to formalize the notion of a landscape and to make the result as general and useful as possible for thinking about computational search.

- **Why the interest in landscapes?**
  My interest in the landscape metaphor stems from an interest in evolutionary algorithms. In the community of researchers working on these algorithms, the word "landscape" is frequently encountered—both informally, in conversations and presentations, and formally, in conference and journal papers. It is natural therefore that one should seek to understand what is meant by the word. Remarkably, it proved difficult to find out exactly what people meant by the word, and when that could be discovered, the definitions did not concur. In my opinion, the landscape metaphor is a powerful one and because its meaning is appears intuitively clear, people are inclined to employ it rather casually. Questions that would be asked of a less familiar and less evocative word simply do not get asked. After all, everyone knows what a landscape is. Or do they? If the term was only used in an informal sense or for the purposes of rough illustration, the situation would be more tolerable. Instead, properties of fitness landscapes are used as the basis for explanations of algorithm behavior and this is often done in a careless fashion that is detrimental to our understanding. Thus, the motivation for trying to establish just what is (or should be) meant by a landscape comes from (1) the fact that the word is in widespread use, (2)

the belief that it is usually not well-defined, and (3) the belief that this lack of definition has important consequences. I will take the first of these as given and argue below that the two beliefs are justified.

- **What is wrong with the current definitions of landscapes?**
  The following are some of the current problems with the use of the landscape metaphor *in the field of evolutionary computation*. The use of landscapes in other fields will be discussed in Chapter 6. Some of the terminology used, (e.g., *operator*), will not be defined until later in the chapter.

  1. In many cases, landscapes are not defined at all. This is very common informally, but also happens quite frequently in formal settings. It is also common for papers to reference an earlier paper that contains no definition or a poor one. This practice has decreased in the last year.
  2. Another common problem is vague definitions of the word. For example, one encounters statements such as "the combination of a search space S and a fitness function $f : S \to \mathbb{R}$ is a fitness landscape." Apart from the fact that what has been defined here is simply a function, it might not be immediately clear what is wrong with such a definition. The problem is that virtually every term we would like to use in describing a landscape depends on being able to define neighborhood but the above definition tells us nothing about neighborhood. For example, perhaps the most frequently used landscape-related term is "peak." But what is a peak? A simple definition is: A point whose neighbors are all less high than it is. It is not possible to sensibly define the word "peak" without defining what constitutes neighborhood. The need to be specific about neighborhood is not widely attended to. As a result, in many cases, it is not clear what is meant by the word landscape. The expressions "fitness function" and "fitness landscape" are often used interchangeably, both formally and informally. A fitness *function* is a function: a mapping from one set of objects to another. There is no notion of neighborhood, and one is not needed. A fitness *landscape* requires a notion of neighborhood.
  3. Even when neighborhood is defined, it is often done so incorrectly. Informally, people will staunchly defend the binary hypercube as *the* landscape when an algorithm processes bit strings, even if the algorithm never employs an operator that always flips exactly one bit chosen uniformly at random in an individual. Certainly this hypercube may have vertices that fit the above definition of a peak, but the relevance of these points to such

an algorithm is far from clear. Even when mutation is employed in a GA, the mutation operator does *not* induce a hypercube graph. The mutation operator used in a GA does not always alter a single bit in an individual and hence the mutation landscape graph is not a hypercube. For binary strings of length n, the graph is the complete graph K_{2^n} augmented with a loop from each vertex to itself. With a per-bit mutation probability of p, the bidirectional edge between two vertices, v_1 and v_2 with Hamming distance d has a transition probability of $p^d(1-p)^{n-d}$. This is the probability that a single application of the mutation operator transforms v_1 into v_2 (or vice-versa). Figure 3 on page 33 illustrates such a graph.

4. Even though neighborhood is sometimes specified, and specified correctly, a single landscape is very often used as the framework for considering the operation of an algorithm that uses multiple operators. The best example is the use of a hypercube graph landscape as a basis for consideration of both mutation and crossover in a GA. Apart from the fact that the hypercube is not induced by either operator, this position leads to statements such as "crossover is making large jumps on the landscape." Crossover is viewed here in terms of the landscape structure that mutation is popularly thought to create. Mutation takes single steps, but crossover causes large jumps. Why is this? Even when the mutation probability in a GA is set to zero, people will insist on talking about local maxima from the point of view of mutation, despite the fact that these clearly have no relevance whatsoever to the algorithm. The model presented in this chapter claims that each operator is taking single steps on its own landscape graph. It is possible to consider the effect of crossover in terms of the mutation graph, but it is equally possible to consider mutation in terms of the crossover graph. The model insists that we explicitly acknowledge this, rather than, as is now common, comparing one operator in terms of the structure defined by another.

5. In the study of GAs, there is a tremendous bias towards thinking purely in terms of a mutation landscape. This bias can be difficult to detect in oneself. A good illustration comes again from the consideration of the word "peak." As discussed above, for something to be a peak, it must have neighbors. If we define neighborhood in terms of the operators we employ, it should seem strange that we can use the word peak for one operator but not for another. Why is there no notion of a peak under crossover? Why is the term reserved solely for mutation? A GA creates new individuals

through crossover and through mutation. In some instances, the number of individuals created via crossover will be greater than the number created through mutation. In these cases, shouldn't we be more concerned about peaks (local optima) with respect to crossover than with respect to mutation? In fact, if the domain of a fitness function has a point whose fitness is higher than that of all others, the mutation landscape will always have a single global optimum and no local optima, as any point can be reached from any other via one application of the mutation operator. By this reasoning, we should not be concerned with local optima from mutation's point of view, since they do not exist! The current notions of landscape pay no heed to operator transition probabilities, and this is clearly important. The model of landscapes presented in this chapter incorporates these probabilities and removes the mutation landscape bias, giving each operator its own landscape, complete with peaks.

6. The current models of landscape cannot be simply extended to consider other search algorithms. It is not even possible to use them consistently across algorithms within evolutionary computation. Current models typically require fixed dimensionality and a distance metric. This makes it difficult to describe, for instance, the landscape that genetic programming (GP) is operating on. The proposed model makes it simple to view GP as occurring on landscapes, as well as making connections to many search algorithms in other fields, such as those in Artificial Intelligence [38, 39].

2.2. Preliminary Definitions

This section defines multisets and gives a very brief introduction to relations, orders and the theory of directed graphs.

2.2.1. Multisets

A multiset is

> "...a mathematical entity that is like a set, but it is allowed to contain repeated elements; an object may be an element of a multiset several times, and its multiplicity of occurrences is relevant."
>
> Knuth [40, page 454].

For example, $\{1, 1, 4, 5, 5\}$ and $\{4, 1, 5, 5, 1\}$ are equivalent multisets. In contrast, $\{1, 1, 2\}$ and $\{1, 2\}$ are equivalent as sets, but not as multisets. Of course, any set is also a multiset. Binary operations $\cup$, $\uplus$ and $\cap$ between multisets that obey commutative, associative, distributive and other laws can be simply defined [40, page 636], though we will have no use for them in this dissertation. Knuth mentions a number of situations in which multisets arise naturally in mathematics. Kanerva makes extensive use of multisets in his analysis of Sparse Distributed Memory [41]. Given a set S, I will use $\mathcal{M}(S)$ to denote the infinite set of all multisets whose elements are drawn from S. Thus the multiset above is an element of $\mathcal{M}(\{1,4,5\})$, as are the multisets $\{1\}$, $\{4,4,4,4\}$, $\{4,4\}$ and $\{1,4,5\}$ itself. I will use $|S|$ to represent the number of elements in the multiset S. Hence, $|\{1,5,5\}| = 3$. Finally, define

$$\mathcal{M}_q(S) = \{s \in \mathcal{M}(S) : |s| = q\},$$

as the set of multisets of S with cardinality q.

2.2.2. Relations, Orders and Digraphs

A *relation*, R, between two sets A and B is a subset of $A \times B$. That is, $R \subseteq \{(a,b) \mid a \in A \text{ and } b \in B\}$. The notation aRb is used to indicate that $(a,b) \in R$. When $A = B$, R is said to be *reflexive* if $aRa\ \forall a \in A$; *symmetric* if $aRb \Rightarrow bRa\ \forall a,b \in A$; *transitive* if $aRb, bRc \Rightarrow aRc\ \forall a,b,c \in A$ and *complete* if, given $a \neq b$, at least one of aRb or bRa is true. A relation that is not complete is said to be *partial.* Different combinations of the above four properties impose different "orders" on A. If R is irreflexive, asymmetric, transitive and complete, it is called a *total order* or *complete order.* If it is irreflexive, asymmetric and transitive, but not complete, R is a *partial order.* The relation "greater than" is a total order on the set of real numbers. The relation "is an ancestor of" is a partial order on the set of all people. In this dissertation we will commonly be concerned with partial orders that are used to rank potential solutions to search problems. A relation that is reflexive, symmetric and transitive is called an *equivalence* relation. An equivalence relation creates *equivalence classes*: If R is an equivalence relation and aRb then a and b are members of the same equivalence class.

A relation R on a set A can also be thought of as a directed graph or *digraph.* The vertices of the digraph are the elements of A and there is a directed edge from vertex a to vertex b if aRb. It is traditional to exclude the possibility of edges from a vertex to itself in a digraph. This is mainly a matter of convention, and we shall allow such edges or *loops.* If R is both symmetric and complete, the digraph contains all possible edges and is denoted K_n when the digraph has n vertices. A symmetric digraph can also be regarded as an undirected *graph.* In this case, the two directed edges between a pair of vertices can be drawn as a single undirected edge. In this dissertation, we shall simply use the word "graph" to mean a directed graph. We will commonly discuss digraphs that correspond to symmetric relations and the distinction will be unimportant.

Finally, a pair of vertices v and w are *connected* if there is an undirected path between them. That is, w can be reached by traversing edges (irrespective of the directions on the edges) from v. If v is connected to w, w is therefore connected to v. The difficulty with edge directions disappears if the underlying relation is symmetric.[1] If every vertex is considered connected to itself, the relation "is connected to" is an equivalence relation on the set of vertices and thus induces equivalence classes. Each of these equivalence classes, together with the edges incident on the vertices in the class, defines a *connected component* of the graph. If a graph has a single connected component, it will be said to be *fully connected* or simply *connected*. Otherwise, the digraph is a *forest* of connected components and is said to be *disconnected*. For a much fuller discussion of connectedness in digraphs, see Harary, Norman and Cartwright [42], which also provides a good introduction to the discussion of graphs in this dissertation.

2.3. Search Problems and Search Algorithms

As mentioned in the introduction to this chapter, it will be assumed that we are interested in solving a search problem. This position has been adopted by others and, given a general interpretation of search, is not terribly restrictive. The following quotes lend support to this claim:

> "In its broadest sense, problem solving encompasses *all* of computer science because any computational task can be regarded as a problem to be solved." Nilsson [43, page 2].

> "Every problem-solving activity can be regarded as the task of finding or constructing an object with given characteristics." Pearl [31, page 15].

If we accept these statements, and consider the problem of "finding or constructing an object" a search problem, then we should not be surprised if very many situations

[1]This is a simplistic definition of connectedness in a digraph. However, it is sufficient for our purposes. A better treatment of this issue can be found in Harary, Norman and Cartwright [42].

can be thought of as posing search problems. As a final link in this chain of reasoning, Rich claims that

> "Every search process can be viewed as a traversal of a directed graph in which each node represents a problem state and each arc represents a relationship between the states represented by the nodes it connects."
>
> Rich [44, page 55].

If we take these three excerpts seriously, search is not only ubiquitous, but it can be described in terms of a process on a graph structure. On setting out to develop a model of fitness landscapes, I soon arrived at the conclusion that a landscape could be conveniently viewed as a graph. Some time later, it became apparent that this was a common view of search in other fields and I took pains to ensure that the model was general enough to be appropriate for discussions of these algorithms also. Given such a broad target, it is necessary to develop the model in a very general way. I am particularly concerned that this be done in a mathematically precise way that will allow formal definitions and discussion when that is needed. This is similar to the aims expressed by Banerji [45].

In setting out to present a model of computational search, there are several fundamental issues that need to be initially addressed. I assume, in line with the quotation due to Pearl above, that a search problem takes the form "Find an object with the following properties...." Then search is conducted amongst a (possibly infinite) set of objects (potential solutions). As was emphasized in §1.3(5), the first step in solving a search problem involves making a choice about the contents of this set. I will call this set of objects the *object space*, and it will be denoted by $\mathcal{O}$. This choice will often be a simple one and in many cases a good candidate for $\mathcal{O}$ will be provided in the problem statement itself. This decision is typically made before a computer is introduced as an aid to search. Example object spaces include the set of n-tuples of real numbers, the set of binary strings of length n, the set of LISP S-expressions, the set of permutations of the integers one to ten, the set of legal chess positions, the set of spin configurations in a spin glass and the set of RNA sequences.

A second step that is common when searching is to decide on some representation of the objects in $\mathcal{O}$. The representation determines a set, $\mathcal{R}$, the *representation space*, that can be manipulated more conveniently than $\mathcal{O}$ by the searcher. The model of search of this chapter does not require that $\mathcal{R} \neq \mathcal{O}$, but this will be true throughout this dissertation. In situations where $\mathcal{R} = \mathcal{O}$, (i.e., no representation is employed), one can substitute $\mathcal{O}$ for $\mathcal{R}$ in the model. When performing search via a computer, $\mathcal{R}$

will very commonly be the result of a choice of data structure, $\mathcal{D}$.[†] In this dissertation, this will always be the case. The possible ways of assigning values to the bits that comprise the memory corresponding to an instance of $\mathcal{D}$ determines $\mathcal{R}$. Elements of $\mathcal{R}$ will be used to represent elements of $\mathcal{O}$. Representation via instances of a data structure is not the only possible choice in computational search, it is merely the most common. Forms of search involving analog computation may have representations involving resistors (for example), in which case the representation space may have nothing to do with data structures [46].

Unlike the object space, the representation space is necessarily finite. Computers have finite limits and therefore, in general, it is not possible to represent the whole of $\mathcal{O}$ with $\mathcal{R}$, at any given point in time. Such a situation is very common in evolutionary algorithms. For this reason, only a finite subset $\mathcal{O}' \subseteq \mathcal{O}$ will be represented at any given time. The mapping between elements of $\mathcal{O}'$ and $\mathcal{R}$ will be called the *representation*.[2] The representation is the link between the objects we choose to examine as potential solutions to the search problem and the objects we choose to manipulate to effect the search. When the search is finished, we invert this mapping to produce the solution.

A relation, Γ, ("is represented by") between the sets $\mathcal{O}$ and $\mathcal{R}$ implements this representation. For $o \in \mathcal{O}$ and $r \in \mathcal{R}$, we will write $o\,\Gamma\,r$ to indicate that o is represented by r. The inverse relation Γ^{-1} ("represents") between the sets $\mathcal{R}$ and $\mathcal{O}$ is defined as $\Gamma^{-1} = \{(r, o) \mid (o, r) \in \Gamma\}$. We will write $r\,\Gamma^{-1}\,o$ to indicate that r represents o. The (possibly empty) subset of $\mathcal{R}$ representing $o \in \mathcal{O}$ will be denoted by $\Gamma(o)$ and the (possibly empty) subset of $\mathcal{O}$ that is represented by $r \in \mathcal{R}$ will be denoted by $\Gamma^{-1}(r)$. If $\Gamma(o) \neq \emptyset$, we will say that o is *represented*. If $\Gamma^{-1}(r) = \emptyset$ we will say that r is *illegal*. $\mathcal{R}' = \{r \in \mathcal{R} \mid \Gamma^{-1}(r) \neq \emptyset\}$ will be used to denote the set of elements of $\mathcal{R}$ that are not illegal. Consideration of how an algorithm restricts its attention to $\mathcal{R}'$ or manages excursions into $\mathcal{R} - \mathcal{R}'$ are discussed in §2.12(41). There is nothing to stop an algorithm from adopting a new $\mathcal{O}$, $\mathcal{O}'$, $\mathcal{R}$, $\mathcal{R}'$, or Γ at any point, for example see [48, 49, 50, 51].

At this point, we have almost enough to talk about search. We need to be slightly more specific about the nature of the search problems in which we are interested. As mentioned above, we do not need to be too specific about the details of the search problems if those details do not influence the landscapes that we will construct. In this dissertation I will be concerned with two types of search problems, which I call

[†]This does not restrict us to data structures of fixed size.

[2]Liepins and Vose have termed this the *embedding* [47].

simply *Type 1* and *Type 2*:

1. A *Type 1* search problem requires the location of an object (or objects) possessing a set P of properties. The search is only satisfied when such an object is located. The statement of the search problem includes no notion of "closeness" to a solution. Either a solution has been found and the search is successful, or one has not been found and it is unsuccessful. It is convenient to imagine the statement of the search problem as providing a function $g : \mathcal{O} \rightarrow \{0, 1\}$ defined as follows:
$$g(o) = \begin{cases} 1 & \text{if } o \text{ possesses all properties in } P. \\ 0 & \text{otherwise.} \end{cases}$$
which can be used to determine if an object satisfies the requirements of the search problem.

2. In a *Type 2* search problem, the searcher is required to find as good an object as possible (presumably within some limits on time and computation). Once again, it is convenient to imagine that the statement of the search problem provides a function $g : \mathcal{O} \rightarrow \mathcal{G}$, for some set $\mathcal{G}$, and a partial order $>_{\mathcal{G}}$ on $\mathcal{G}$ such that for $o_1, o_2 \in \mathcal{O}$, if $g(o_1) >_{\mathcal{G}} g(o_2)$ then o_1 is considered a better answer to the search problem than o_2. Notice that in a *Type 2* problem, a searcher that does not exhaust the whole of $\mathcal{R}$ cannot know whether an optimal object has been located (given that choice of $\mathcal{R}$), and hence has no reason to stop searching based on the quality of the objects it has located.

Our search algorithms represent potential solutions to the problem using instances of the data structure $\mathcal{D}$. At point t in time, the algorithm will have memory allocated for a finite set of these, and the values they contain will form a multiset, $\mathcal{C}_t \in \mathcal{M}(\mathcal{R})$. When we are not concerned with time, we will drop the subscript and simply use $\mathcal{C}$ to denote the current multiset of $\mathcal{R}$. In the discussion of landscapes to come, we will be concerned with $\mathcal{C}$ and with the methods used by the algorithm to change it. The algorithms employ *operators* to modify $\mathcal{C}$. In this dissertation, lower case Greek symbols will be used exclusively to refer to operators. In particular, ϕ (and less frequently ψ), will be reserved to indicate generic operators.

An operator is a function $\phi : \mathcal{M}(\mathcal{R}) \times \mathcal{M}(\mathcal{R}) \rightarrow [0..1]$. The value of $\phi(v, w) = p$ for $v, w \in \mathcal{M}(\mathcal{R})$ indicates that with probability p, v is transformed into w by a single application of the stochastic procedure represented by the operator ϕ. Thus, for any operator ϕ and $v \in \mathcal{M}(\mathcal{R})$,

$$\sum_{w \in \mathcal{M}(\mathcal{R})} \phi(v, w) = 1.$$

For convenience, I will talk of the "action" of an operator and mean the action of the computational procedure that the operator represents. Any modification made to $\mathcal{C}$ by the algorithm shall be considered to be the result of the application of a procedure associated with some operator. The series of applications of operators by an algorithm can be illustrated by

$$\mathcal{C}_0 \xrightarrow{\phi_1} \mathcal{C}_1 \xrightarrow{\phi_2} \mathcal{C}_2 \xrightarrow{\phi_3} \cdots.$$

The details of the workings of an operator have deliberately not been specified. As a result, operators are so broad in scope, that we may regard *every* method an algorithm uses to affect $\mathcal{C}$ as taking place through the action of an operator. Thus there are operators that (1) increase the size of $\mathcal{C}$ through the allocation of memory to serve as instances of $\mathcal{D}$, (2) reduce the size of $\mathcal{C}$ through freeing allocated memory, and (3) alter the contents of existing members of $\mathcal{C}$. Some operators may combine any of these three actions. Designating some action of the algorithm as being the result of an operator can be done quite arbitrarily. For instance, if we examine $\mathcal{C}_t$ and later examine $\mathcal{C}_{t+\delta}$, we can imagine that the algorithm has employed some operator that transformed $\mathcal{C}_t$ into $\mathcal{C}_{t+\delta}$ that took δ time units to complete. Depending on the value of δ, we may be talking of the effect of a single microcoded instruction in the CPU or of the action of the entire algorithm.

Clearly, some operators will have more effect on the result of the search than others. Though crucial for the algorithm's operation, an operator that increases the size of $\mathcal{C}$ by one (through memory allocation) will typically be uninteresting, as few algorithms deliberately make use of uninitialized memory. An initialization operator that sets the bits of an element of $\mathcal{C}$ may be somewhat more interesting. What we choose to consider an operator will depend on what we are interested in studying. In this dissertation, we will be concerned with operators such as mutation, crossover and selection in evolutionary algorithms. These represent one natural perspective on the working of these algorithms and reflect an interest in the importance of these operators. These operators are natural, in an informal sense, since they are typically implemented as separate procedures in the programs that implement the algorithms. Another natural perspective is to consider a generation of an evolutionary algorithm to be an operator. Such an operator usually includes some combination of the three finer-grained operators just mentioned. This is the perspective that has (so far) been adopted by researchers interested in understanding aspects of the genetic algorithm through viewing it as a Markov chain (see [52] for example). The set of operators of interest in an algorithm will typically be only a part of the algorithm. The algorithm must also make important decisions about which operators to apply and when, decide what elements of $\mathcal{C}$ should be acted on by operators, and decide when the search should be terminated.

Given an operator ϕ, the *ϕ-neighborhood* of $v \in \mathcal{M}(\mathcal{R})$, which we will denote by $N_\phi(v)$, is the set of elements of $\mathcal{M}(\mathcal{R})$ accessible from v via a single use of the operator. That is, $N_\phi(v) = \{w \in \mathcal{M}(\mathcal{R}) \mid \phi(v, w) > 0\}$. If $w \in N_\phi(v)$ we will say that w is a ϕ-neighbor of v. When the operator in question is understood, the terms "neighborhood" and "neighbor" will sometimes be used. However, it is important to keep in mind that this is an abbreviation. Two points that are neighbors under one operator may not be under another. Given $P \subseteq \mathcal{M}(\mathcal{R})$, we will use $N_\phi(P)$ to represent the set of elements of $\mathcal{M}(\mathcal{R}) - P$ that neighbor an element of P. That is,

$$N_\phi(P) = \{w \in \mathcal{M}(\mathcal{R}) - P \mid \phi(v, w) > 0 \text{ and } v \in P\}.$$

A situation that we will often encounter arises when an operator ϕ replaces an element $v \in \mathcal{C}$ by choosing one of a subset of $N_\psi(v)$. That is, the job of ϕ is to replace v with one member of the neighbors of v as defined by a second operator ψ. An example will make this clearer. Suppose we have a hillclimbing algorithm; that $\mathcal{R}$ is the set of all binary strings of length n; and that $v \in \mathcal{R}$ is the current location of the hillclimbing search. Operator ψ flips a randomly chosen bit in a binary string. Operator ϕ uses ψ some number of times to generate a subset of $N_\psi(v)$. These are placed by ϕ into temporary locations in $\mathcal{C}$. When some number of neighbors of v under ψ have been generated, or when one is found that is suitable, ϕ replaces v with the selected neighbor. Another example of this sort of operator is that which effects a move in a simulated annealing algorithm [53]. It employs a mutation operator to generate neighbors of the current point until it finds an acceptable one and sets the memory corresponding to the current point to the value of the selected neighbor.

This brings us to the final aspect of search algorithms that we will consider before describing landscapes. When an algorithm makes a decision about the relative worth of a set of multisets of $\mathcal{R}$ (such as the neighbors generated in the example above), it must have some basis for this decision. Although the decision can be taken by simple use of a pseudo-random number generator, it is more common that the algorithm will have some way of computing the worth of an element of $\mathcal{M}(\mathcal{R})$. We will suppose the search algorithm has a function $f : \mathcal{M}(\mathcal{R}) \to \mathcal{F}$, for some set $\mathcal{F}$, and a partial order $>_\mathcal{F}$ over $\mathcal{F}$. If $v, w \in \mathcal{M}(\mathcal{R})$ and $f(v) >_\mathcal{F} f(w)$ then the multiset v will be considered in some sense better for the purposes of continuing the search than the multiset w. Examples of such decision-making include the use of selection in an evolutionary algorithms to produce one population from another, pruning methods in dynamic programming and the acceptance test in simulated annealing. In the first two of these examples, the algorithm is effectively making choices about multiple elements of $\mathcal{R}$, rather than about a single element of $\mathcal{R}$. This is the reason that the domain of f is $\mathcal{M}(\mathcal{R})$. We will call $\mathcal{F}$ the *fitness space*.

In all cases, the function f must be chosen by the searcher, it is not part of the problem. It might appear that when solving a *Type 2* search problem that when evaluating multisets of size one, the searcher would always choose $f = g$, $\mathcal{F} = \mathcal{G}$ and $>_\mathcal{F}=>_\mathcal{G}$. While this might be a very natural choice, it is still a choice. Suppose, for example, that the problem is to find a Hamiltonian circuit through a given graph whose edges are labeled with distances. A *Type 2* instance of this problem may specify that a circuit c_1 can be considered better than a circuit c_2 if the sum of the distances on the edges of c_1 is less than the sum on the edges of c_2. However, the searcher might believe that the search might be best approached by looking for circuits that form simple closed curves, and therefore aim to minimize the number of times a circuit crosses itself. Or the number of crossings may be combined with other information, such as the length of the circuit, to form another function that is used for search. Of course, the answer that the algorithm eventually produces will be evaluated only in terms of the function g, but that does *not* mean that the search has to be conducted using it. In simulated annealing and genetic algorithms, it is common to search using an f that incorporates g and some other function that gives penalizes illegal objects (see, for example, [54, 55]). On a *Type 1* search problem, it is more apparent that the searcher needs to devise some measure of worth, since no information is given at all. In this case, the function that is chosen has been called an *heuristic* function in Artificial Intelligence.

2.4. Landscapes

A landscape is dependent on five of the components of search and algorithms discussed in the previous section. We may write a landscape as

$$\mathcal{L} = (\mathcal{R}, \phi, f, \mathcal{F}, >_\mathcal{F}).$$

The components are, respectively, the representation space, an operator, the function $f : \mathcal{M}(\mathcal{R}) \rightarrow \mathcal{F}$, for some set $\mathcal{F}$, and a partial order $>_\mathcal{F}$ over $\mathcal{F}$. As emphasized in earlier sections, a landscape is a metaphor by which we hope to imagine some aspect of the behavior of an algorithm. That can be done by viewing the 5-tuple as defining a directed, labeled, graph $G_{\mathcal{L}} = (V, E)$ where $V \subseteq \mathcal{M}(\mathcal{R})$, $E \subseteq V \times V$ and $(v, w) \in E \iff \phi(v, w) > 0$. A vertex $v \in V$ will be labeled with $f(v)$. An edge (v, w) will be labeled with $\phi(v, w)$, the probability that the action associated with the operator ϕ produces w from v. Though a landscape is formally defined by a 5-tuple, we will also talk of a landscape as though it were the graph that arises from the 5-tuple.

The label $f(v)$ attached to a vertex v can be thought of as giving the "height" of that vertex. This is in keeping with the imagery we usually associate with landscapes. This value will often be referred to as the *fitness* of the multiset of $\mathcal{R}$ represented by v. The partial order, $>_{\mathcal{F}}$, is used to determine relative fitness (height). In many cases, $>_{\mathcal{F}}$ will actually be a total order. The out-degree of a vertex in a landscape graph will be the same as the size of the neighborhood of the corresponding multiset. When $\phi(v, w) = \phi(w, v)$ for all $v, w \in V$, we will consider the landscape graph as undirected and draw a single edge between v and w with the understanding that the edge is bidirected. It should be remembered that the vertices of the landscape graph correspond to multisets of elements from $\mathcal{R}$, not to single elements of $\mathcal{R}$. This is important, because it allows us to define landscape graphs for arbitrary operators, not just those that act on and produce a single element of $\mathcal{R}$ (e.g., mutation in a genetic algorithm). Landscapes in this model are well-defined no matter how many elements of $\mathcal{R}$ the operator acts on or produces, even zero. It will also be important to remember that each operator employed by a search algorithm helps create a landscape graph. Thus if an algorithm employs three operators, it can be thought of as traversing edges on three graphs.

2.5. Definitions and Special Cases

This section provides definitions for some of the terms that are commonly encountered in discussions of landscapes. In all of the following definitions, we assume that a landscape $\mathcal{L} = (\mathcal{R}, \phi, f, \mathcal{F}, >_{\mathcal{F}})$ is given. Some of the following terminology is most useful when $>_{\mathcal{F}}$ is a total order. For example, if $>_{\mathcal{F}}$ is not complete, given v and $w \in N_{\phi}(v)$, it may be that neither $v >_{\mathcal{F}} w$ nor $w >_{\mathcal{F}} v$ is true. In this case, we cannot decide if either vertex is a peak.

2.5.1. ϕ-neighborhood and ϕ-neighbor

These important terms have already been introduced as they were necessary for the presentation of the model. They are defined in §2.3(24).

2.5.2. ϕ-maximum or ϕ-peak

A *ϕ-maximum* or *ϕ-peak* is a vertex

$$v \in V \mid \forall w \in N_\phi(v), f(v) >_{\mathcal{F}} f(w).$$

In words, a vertex is a ϕ-peak if its fitness value is greater than the fitness values of all its neighbors under the operator ϕ. When we are not concerned with any operator in particular, we will simply talk of peaks. Notice that a peak is a vertex in a landscape, not a point in $\mathcal{R}$ or $\mathcal{O}$. In some cases, e.g., with mutation operators, the vertex will correspond to a single element of $\mathcal{R}$, but in others, e.g., with crossover operators, it will not. This definition allows every operator to have peaks, instead of reserving the notion for a single-change (see §2.6.1(31)) or mutation operator. Consequently, a vertex that is a peak from the point of view of one operator is not necessarily a peak from the point of view of another. This is the reason for calling it a *ϕ-peak* instead of merely a *peak*. This terminology is more awkward, but the point is an important one.

2.5.3. global-maximum

A *global-maximum* or *global-optimum* of a landscape is a vertex

$$v \in V \mid \forall w \in V, f(v) >_{\mathcal{F}} f(w).$$

That is, a vertex is a global-maximum if it is at least as fit as every other vertex. If a vertex has maximal fitness, then it will be a global maximum no matter who its neighbors are. For this reason, we can call a vertex a "global maximum" without an operator prefix. If a vertex is a global maximum under one operator, then it will also be under all other operators.

2.5.4. ϕ-local-maximum

A *ϕ-local-maximum* or *ϕ-local-optimum* is a ϕ-peak that is not a ϕ-global-maximum. In this case, unlike with global maxima, the ϕ prefix is important. A vertex v that is a ϕ-local-maximum will usually not be a ψ-local-maximum as, in general, $N_\phi(v) \neq N_\psi(v)$.

2.5.5. ϕ-plateau

A *ϕ-plateau* is a set of vertices

$$M \subseteq V, |M| > 1 : \forall\ v_0, v_n \in M,\ \exists v_1, \ldots, v_{n-1} \text{ with}$$
$$f(v_i) = f(v_{i+1}) \text{ and } v_{i+1} \in N_\phi(v_i) \quad \forall\ 0 \leq i < n.$$

This is a connected set of at least two vertices that all have the same fitness. It is possible to move using ϕ between any two vertices visiting only vertices with equal fitness. Our usual three-dimensional image of a plateau involves a flat area in which, for the most part, a step in any direction will not result in a drop in altitude. This is not the case in the above definition, and it is important to keep this in mind. For example, a ϕ-plateau may surround areas of exceptionally low fitness, as does the rim of a volcano. In such cases, there may be no points on the plateau that have the same fitness as all their neighbors. Defining a plateau to better fit our intuitions is difficult and besides, what we will most often be dealing with, at least in the chapters that follow, will be a ϕ-plateau as defined here.

2.5.6. ϕ-mesa

A ϕ-mesa is a ϕ-plateau, P, whose points have fitness k such that

$$\forall v \in N_\phi(P), f(v) <_{\mathcal{F}} k.$$

That is, a ϕ-mesa is a ϕ-plateau with the additional property that no point on the plateau has a neighbor of higher fitness. Such a set is a connected region of the landscape in which an algorithm that only moves to vertices of equal or higher fitness may wander indefinitely without making any improvement. A ϕ-mesa of size one is also a peak.

2.5.7. ϕ-saddle-region

A saddle point is a notion that is usually associated with continuous spaces. It is not immediately clear how to define a saddle *point* in a high-dimensional discrete space. If it really is to be a point and it is a saddle in the normal sense, then one definition would simply require the point to have at least one uphill neighbor and at least one downhill neighbor. As this will be true of most points, the flavor of a saddle point in a real space is lost. The term becomes practically useless since the only points that are not then saddle points are peaks (maximal or minimal).

A better definition is to use *ϕ-saddle-region* to refer to a ϕ-plateau that is not a ϕ-mesa. This is a region rather than a point. This definition also loses an important part of the flavor of a saddle point, but at least it is useful and the name change highlights the loss.

2.5.8. ϕ-basin-of-attraction

The ϕ-basin-of-attraction of a vertex v_n is the set of vertices

$$B_\phi(v_n) = \{\, v_0 \in V \mid \exists v_1, \ldots, v_{n-1} \text{ with}$$

$$v_{i+1} \in N_\phi(v_i) \quad \forall\, 0 \le i < n \,\}.$$

Thus the basin of attraction of a vertex v is the set of vertices from which v may be reached using ϕ. Notice that $w \in B_\phi(v) \not\Rightarrow v \in B_\phi(w)$.

2.5.9. Fixed Cardinality Operators

Given $\phi : \mathcal{M}_k(\mathcal{R}) \times \mathcal{M}_l(\mathcal{R}) \to [0..1]$ for a fixed k and l, then we shall say that ϕ is a *fixed cardinality* operator. Otherwise, ϕ will be said to be of *variable cardinality.* We will use $\phi_{k \to l}$ to represent the set of all fixed cardinality operators with domain $\mathcal{M}_k(\mathcal{R}) \times \mathcal{M}_l(\mathcal{R})$. Fixed cardinality operators will be treated so commonly in what follows, that an unqualified use of the word "operator" implies that the operator has fixed cardinality.

2.5.10. Walkable Operators

Fixed cardinality operators that are members of $\phi_{k \to k}$ will be called *walkable.* If an operator is walkable, the output of the action corresponding to the operator can be used as its next input. For example, the common forms of mutation are always walkable. Any form of crossover that takes two parents and produces two children is walkable. Crossover that produces one child from two parents is not walkable.

2.5.11. Symmetric Operators

An operator ϕ will be called *symmetric* if $\phi(v, w) = \phi(w, v)$ for all vertices $v, w \in V$. That is, an operator is symmetric if the probability of it producing v from w is the same as the probability of it producing w from v. Notice that if this condition is true, then so are the implications:

$$w \in N_\phi(v) \Rightarrow v \in N_\phi(w) \quad \text{and} \quad w \in B_\phi(v) \Rightarrow v \in B_\phi(w).$$

Hence a symmetric operator is necessarily walkable. Most common operators in evolutionary algorithms are symmetric. As mentioned in §2.4(25), if an operator is symmetric it will be convenient to consider $G_\mathcal{L}$ as undirected, in which case edges may be considered bidirectional and will not be drawn with arrowheads.

2.5.12. ϕ-connected-components

If an operator is symmetric, we will often talk about the ϕ-connected-components of a landscape graph. Two vertices $v, w \in V$ are ϕ-connected in $G_\mathcal{L}$ if there is at least one path between them (irrespective of edge directions). Following the definitions of connectedness in §2.2.2(18) leads naturally to ϕ-connected landscapes. The operator associated with a connected landscape is necessarily walkable, but the converse is not true, as will be demonstrated in Chapter 3. If

$$\phi(v, w) = \phi(v, x) \quad \forall v \in V \text{ and } \forall w, x \in N_\phi(v)$$

we will usually not label the edges of $G_\mathcal{L}$ with the transition probabilities. Figure 1 illustrates a situation where transition probabilities are not equal.

2.5.13. Natural Landscapes

The landscape induced by a fixed cardinality operator will be called *natural* if the operator is symmetric and an element of $\phi_{1\to1}$. Most mutation operators generate natural landscapes. The name is chosen as this is what people usually refer to as a landscape.

2.6. Four Operator Classes and Their Landscapes

This section presents four familiar operators, single-change, mutation, crossover and selection, and briefly describes some properties of the landscapes they generate. These

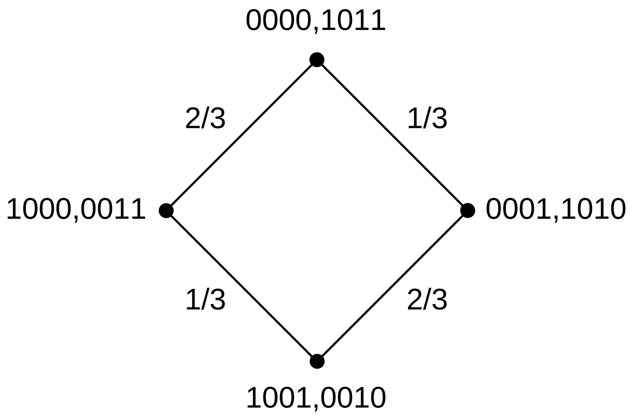

Figure 1. A fragment of the one–point crossover landscape for binary strings of length 4, showing unequal transition probabilities. The operator is symmetric, so the edges of the graph should be considered bidirectional.

operators will be treated so commonly in the remainder of this dissertation that we will give them special names. Three of the four classes of operators most commonly encountered in evolutionary algorithms rely on the objects in $\mathcal{R}$ being composed of some (not necessarily fixed) number of components. For example, a bit string has some number of bits, an n-tuple of real numbers consists of n individual reals and the parse tree representing a LISP S-expression consists of some number of internal nodes and leaves. These components can typically be modified individually or in subsets, possibly according to some constraints.

2.6.1. Single-change Operators

The simplest class of operators I will consider, a subset of $\phi_{1\to 1}$, act on and produce a single element of $\mathcal{R}$. These operators randomly choose a single component of their input and modify it in some way. The simplest example of this sort of operator is the operator that flips a bit chosen uniformly at random in a bit string, which we will denote by β. Note that the outcome of this operator *always* differs in exactly one component from the input. If the elements of $\mathcal{R}$ have n binary components, β induces a graph that is a hypercube of dimension n. Each edge in the graph has a transition probability of $1/n$. Figure 2 shows the landscape generated by this operator on binary strings of length three.

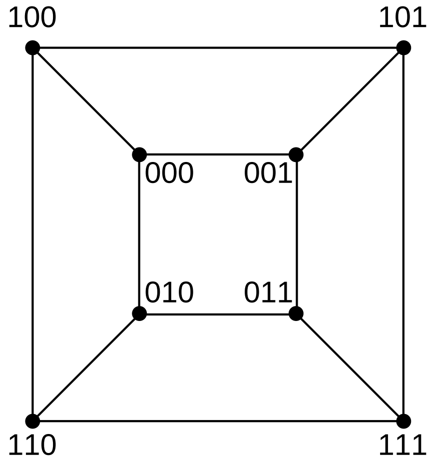

Figure 2. The landscape for the bit-flipping operator (β) on binary strings of length three. Edges are bidirectional and each has probability one-third.

2.6.2. Mutational Operators

The mutational operators, also a subset of $\phi_{1\to 1}$, modify each component of their input with a given, typically small, probability. Thus a mutational operator's output is a modification of its input according to some probability distribution. The outcome may be identical to the input, it may contain minor changes or may even be different from the input in every component. We will denote a generic mutational operator by μ. If we again consider binary strings of length n and assume that each bit is flipped with a fixed probability p, then the graph induced is the complete graph K_{2^n}, augmented with a loop at each vertex. An edge in the graph between vertices that represent binary strings whose Hamming distance is d will be labeled with a probability $p^d q^{n-d}$, where $q = 1 - p$. Notice that the structure induced by this operator is quite different from the one generated by the conceptually similar bit-flipping operator. Figure 3 shows the mutation landscape for binary strings of length three. The operators in both of these classes are walkable. The hypercube induced by β operator will, of course, be a subgraph of the graph generated by μ. If a mutation operator induces a complete graph (possibly augmented with loops), then this graph will always contain exactly one mesa, whose points will have maximal fitness. If no two points in the space have the same fitness, this mesa will be of size one and hence a peak.

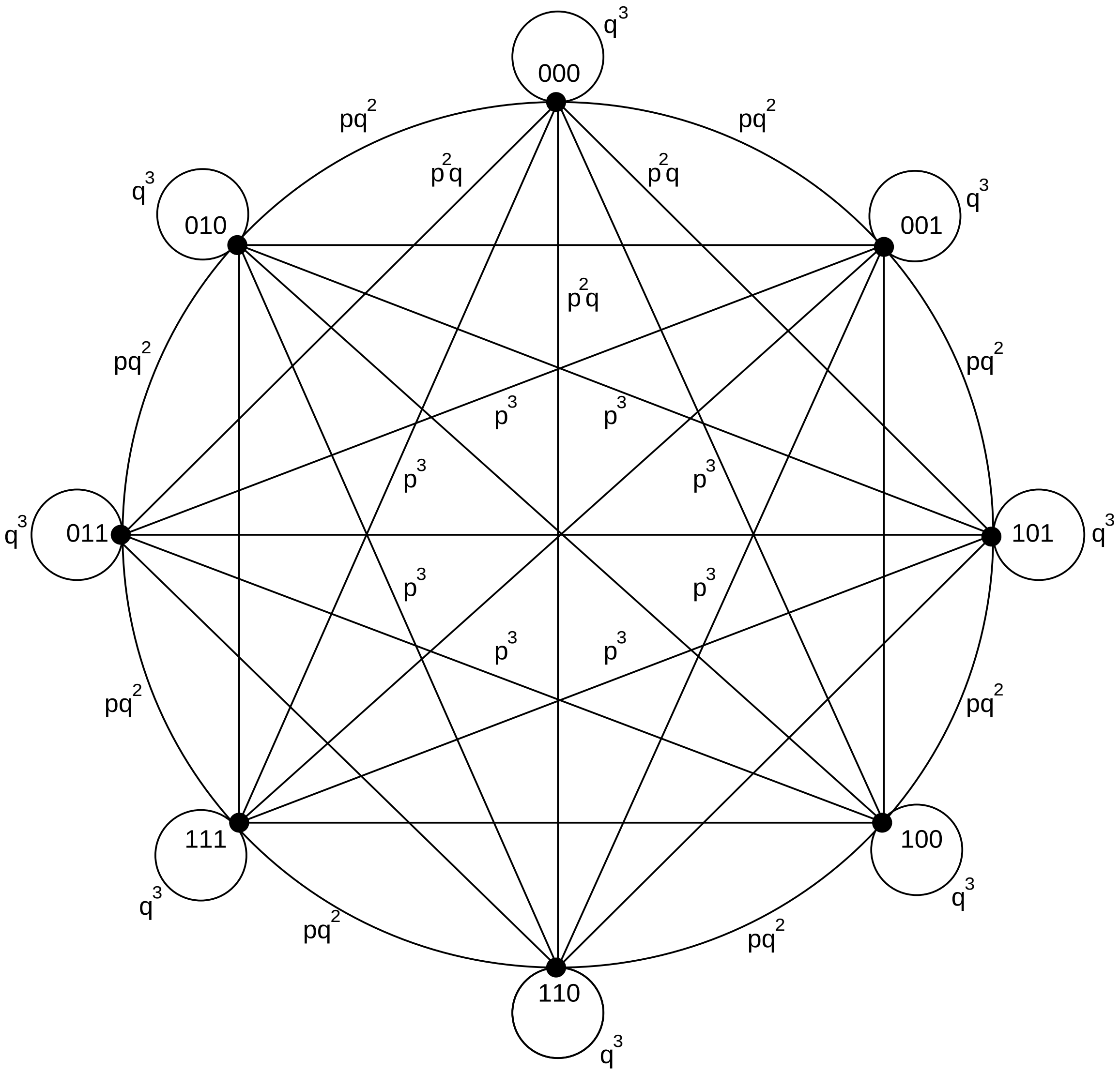

Figure 3. The mutation landscape for binary strings of length three. The mutation probability is p, and $q = 1 - p$. Some edge probabilities are omitted. Edges are bidirectional.

2.6.3. Crossover Operators

The third class, the crossover operators, typically act on an element of $\mathcal{R}^2$. Crossover operators combine the components from their inputs to produce their output. As an example, in one form of crossover in an evolution strategy, the operator receives two n-tuples of real numbers as input and produces a single n-tuple whose elements are the averages of the corresponding components of the inputs [56]. In GAs, the original form of crossover, as described by Holland [9], was *single-point* crossover, which has become more frequently known as *one-point* crossover. There are many forms of crossover for representation spaces such as S-expressions [14, 57, 15], n-tuples of reals [56], permutations of integers [58, 59] and fixed length strings over some alphabet [5]. Figure 4 shows the landscape generated by one-point crossover on binary strings of length three. A generic crossover operator will be denoted by χ. To indicate that a crossover operator acts on k elements of $\mathcal{R}$ and produces l elements of $\mathcal{R}$, a superscript will be added: $\chi^{k\to l}$. A subscript will be used to indicate the common types of crossover: 1 for one-point, 2 for two-point etc., and u for uniform crossover. Thus $\chi_1^{2\to 2}$ is the crossover operator that produces two offspring from two parents using one-point crossover.

2.6.4. Selection Operators

The fourth class is the selection operators. A selection operator acts on a population of individuals (i.e., an element of $\mathcal{M}(\mathcal{R})$) and produces a population. In GAs and their derivatives, selection produces a population whose size is the same as the size of the input population. In some ESs, selection produces a smaller population. For example, in a $(\mu{+}\lambda){-}$ES, the selection operator is a member of $\phi_{\mu+\lambda\to\mu}$. The selection operators exercise two generalities of the model that have not yet been shown to correspond to anything. First, if we regard selection as being an operator that acts on a population of any size, then it is a variable cardinality operator. The second is that selection operators produce landscapes that are not symmetric. Typically, the higher $\phi(v, w)$ is, the lower $\phi(w, v)$ will be, since selection, almost by definition, tends to increase the number of copies of the fittest individuals and decrease the numbers of the less fit. As a result, the above probabilities would very likely only be equal for populations with highly uniform fitness distributions.

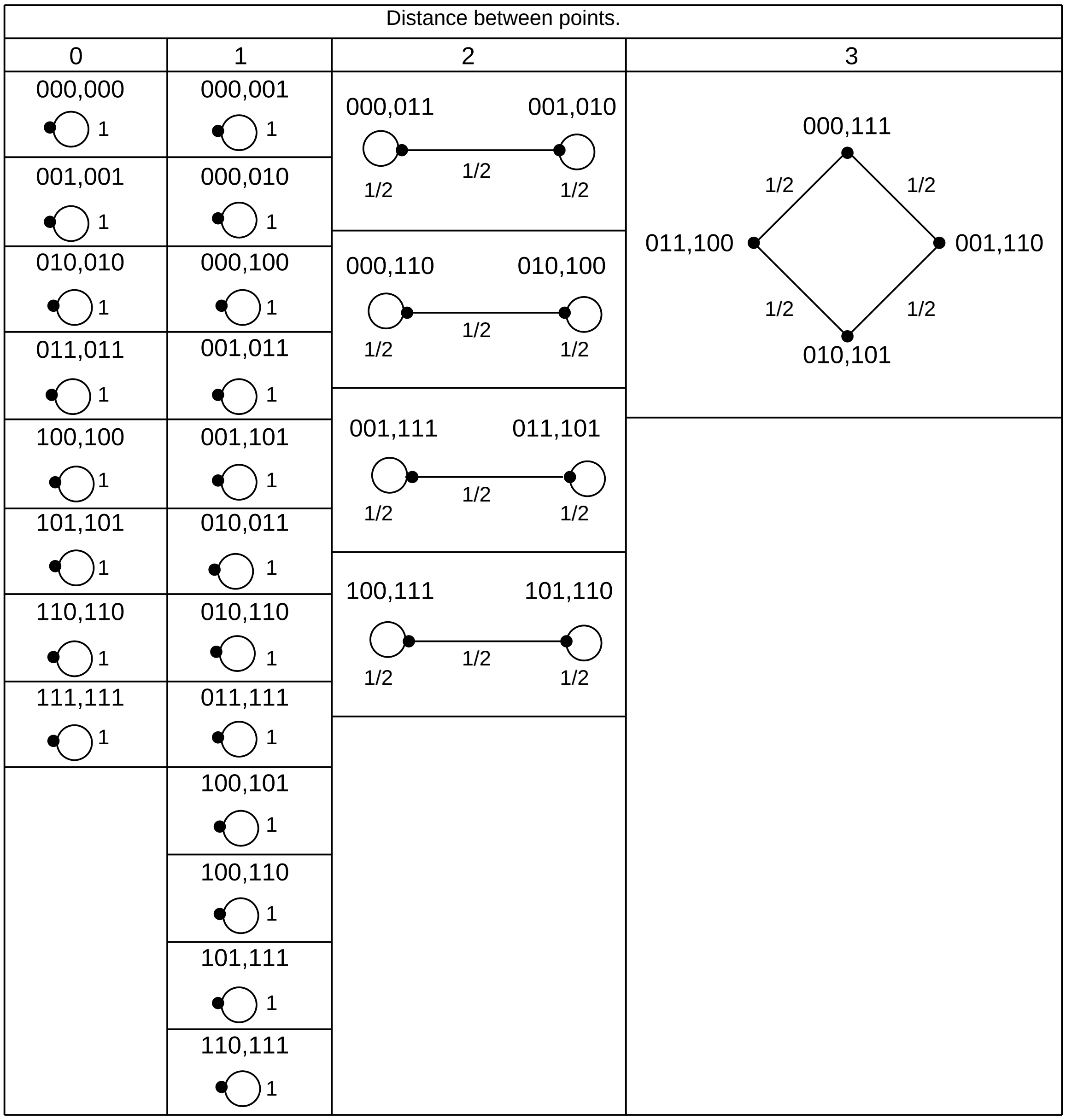

Figure 4. The one-point crossover landscape for binary strings of length three. The crossover operator, $\chi_1^{2\rightarrow 2}$, produces two offspring from two parents. Edges are bidirectional.

2.7. Consequences of the Model

The model presented above has a number of consequences that are not found in other landscape models. Two of these may seem particularly strange. First, as described above, a landscape may not be *walkable*. As mentioned in §2.5(26), an example is the landscape induced by any form of crossover that produces one child from two parents. Such an operator cannot be used to conduct a walk on the landscape as the output of the operator cannot be used as the next input.

Second, landscapes may not be connected. A simple example is the landscape induced by a (non-GP) crossover operator that produces two children from two parents. Consider a vertex of the landscape that corresponds to two points of $\mathcal{R}$ that are identical. The vertex will be connected to itself (with probability 1), and nothing else. A more specific and less trivial example of a landscape that is not connected is seen by considering the vertex $(011, 010)$ of V where $\mathcal{R} = \{0, 1\}^3$. Clearly no form of crossover can transform this input to a pair of points either of which begins with a one, e.g., $(100, 000)$. Equally clearly, no composition of crossovers can accomplish this either. Thus $(011, 010)$ is not connected to any vertex in the landscape which contains a member of $\mathcal{R}$ that starts with a 1 (in fact, this vertex is also connected only to itself). Several complete crossover landscapes are presented in Chapter 3.

The possibility that landscapes may not be connected means that on these landscapes there is no general notion of distance between landscape vertices. This does not mean that a metric cannot be defined, just that there may be no natural one (such as the length of the shortest path between the vertices concerned). The model does not require a distance metric to exist, though the absence of one might make certain statistics meaningless or impossible to compute. Within a connected component of a landscape, one can always use the length of the shortest path between two vertices as a distance metric. The subject of distance metrics and dimensionality is addressed in §2.9(39).

Because a landscape may not be walkable, general statistics describing properties of landscapes may be restricted to using the information gained from repeated single applications of the operator that generated the landscape. This sort of statistic was employed by Manderick et al., even though they were considering a walkable landscape [60].

2.8. Implications for Genetic Algorithms

An important consequence of this model of landscapes is that every operator used by an algorithm defines its own landscape. For this reason, genetic algorithms can be seen as operating on several landscapes; a mutation landscape, a crossover landscape and a selection landscape. These landscapes will have different characteristics for different problems, and can be studied independently. I will refer to this as the "one operator, one landscape" view.

When one hears people talk of "the landscape," the object of reference is usually the mutation landscape or, more commonly, the hypercube landscape that β induces. Despite the fact that crossover may be employed more frequently than mutation in a GA, it has been assigned a supporting role in the landscape structure defined by mutation. To say that crossover is making large jumps on the landscape (the mutation landscape is implied), is to examine crossover outside its own context. What if the algorithm involved does not even use mutation? What relevance does the mutation landscape have then for the algorithm or for crossover? It makes as much sense to say that mutation is making large jumps on the crossover landscape. Though it is obviously important to consider the effect of using multiple operators in an algorithm, it is also important that when we assess an operator in isolation that we do it in light of the landscape structure it defines.

As was mentioned in §2.1(13), the notion of a peak cannot be divorced from the notion of neighborhood. A vertex in $G_{\mathcal{L}}$ can be a peak under one operator and not under another. The mutation landscape has peaks, the crossover landscape has peaks and the selection landscape has peaks. To regard peaks as something defined only in terms of mutation is simply biased.

Another popular notion is that GAs are somehow making a walk on a landscape. I would argue that it is only rarely that such a claim can be supported. A GA that employs crossover that produces one child from two parents and no mutation whatsoever could hardly be said to be walking. This example illustrates the fact that a GA does not require its operators to generate walkable landscapes. It does not follow that when they do, the GA is walking on them. My view of the basic cycle of a GA is of a process that is making moves on three landscapes. Each member of the population corresponds to a vertex in the mutation graph. Some of these are moved, depending on the mutation probability. Then the population members are somehow paired, which defines a set of vertices on a crossover landscape. Many of these vertices make a move on this landscape, depending on the crossover probability. Finally, the entire population corresponds to a point on the selection landscape and a step is taken there before the cycle repeats. This process is depicted in Figure 5. Under this view, the GA is taking single steps on the various landscapes but cannot be said to be walking on any of them. In some cases, obviously, a point may survive many such cycles and take several steps on the mutation landscape. In other cases the population might be highly converged, in which case crossover and mutation will be producing little change and it could be argued that some or even most of the population is walking on the mutation landscape or that the entire population is walking on the selection landscape. But these cases are the exception rather than the norm.

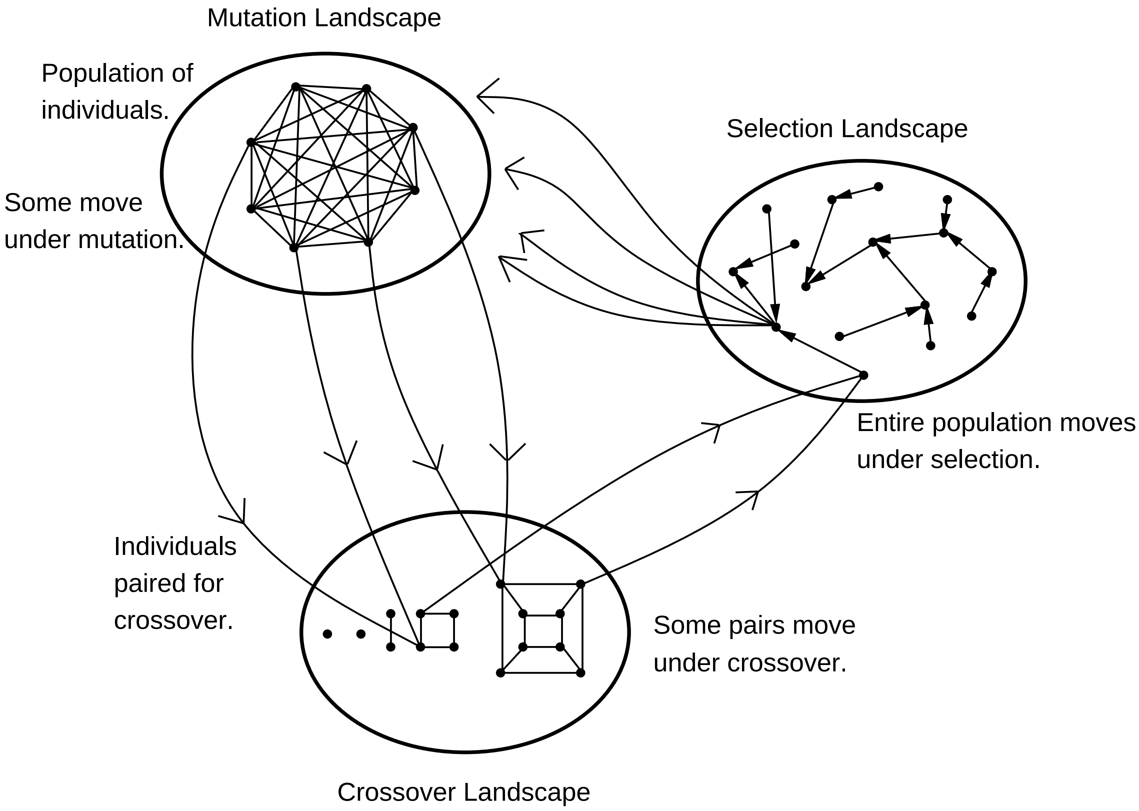

Figure 5. A simplified view of a GA operating on three landscapes. The landscape graphs are idealizations of far larger structures, and self-loops in the graph, created when an operator's input and output are identical, have been omitted. The GA is seen as taking some steps on the mutation landscape, then pairing individuals (probably according to fitness) thereby forming vertices on the crossover landscape upon which moves are made before the entire population is gathered into a vertex on the selection landscape where a step is taken. Finally, the population is decomposed into individuals which again correspond to vertices on the mutation landscape.

2.9. Advantages of the Model

Apart from fact that the model makes it possible to use the landscape metaphor in an unambiguous way, there are several other advantages that deserve mention:

- The model establishes a point of contact with the search algorithms of AI and OR. In all cases the underlying structures being searched are graphs. This relationship is explored more fully in Chapter 5.

- The model does not make the assumptions that prevent other notions of landscape from being more widely used. For example, the landscape does not need some fixed dimensionality, nor does it need a distance metric between the objects that compose the landscape. For these reasons, it is possible to view many AI problems as being problems on landscapes of this type. For instance, one could conceive of a landscape for chess or Rubik's cube. The navigational task on these landscapes may differ, or the operators may be complex, but the underlying structures are the same. The model is also useful within the field of evolutionary computing. It applies as well to GP as to GAs. Statistics that can be calculated for a landscape in one paradigm can be calculated in exactly the same way for another. The model also provides a framework for thinking about HC, SA, ESs and EP.

- The model invites a point of view that seems uncommon in the field of evolutionary computation, though not in AI. This is a view of search as navigation and structure. Once we view search in this way and identify the various components present in a GA, it is natural to ask questions about them. This division and the recombination it makes possible are examined in detail in Chapter 3.

- The "one operator, one landscape" view reveals the very different landscapes that are constructed by various operators. This invites statistical analysis of the landscapes, as will be seen in Chapter 5 and as has been done in [60, 61, 62, 63, 64]. Such analysis has the advantage of being independent of any particular navigation strategy. For this reason, it may be possible to demonstrate that a particular operator creates difficult (in some sense) landscapes for some problem types. Such results might go a long way towards resolving debates on the virtues of certain operators. Statistics such as these would be very useful as indicators of potential difficulty (or ease) of a problem for an operator.

2.10. Limitations of the Model

It can be argued that the model is not particularly useful in situations where the nature of the operators changes in the course of search. For example, the change of mutation vector for an individual in ES and EP, the change of edge probabilities when the temperature falls in simulated annealing or after inversion in a GA, or where the mapping between objects and representation space is changed, e.g., in dynamic parameter encoding [49], delta coding and delta folding [48]. There is some truth to this argument. However, if one ceases to regard a search structure as something necessarily fixed for all time, the model is still potentially useful. For example, a statistic, say correlation length, might be computed for the landscapes generated by a range of different temperature settings in a simulated annealing problem. This might provide useful information about when the search could be expected to make good progress (thus guiding the choice of cooling schedule) and it might prompt a comparison with a hill climber or other algorithm on one of the landscapes. These situations make the landscape something of a moving target, but the targets can be studied individually.

More generally, an algorithm might change many aspects of its behavior, for example the fitness function or the representation space, mid-run, and thereby shift its attention to new landscapes. This might be done very frequently. In these cases, the landscapes model of this dissertation holds that each such change produces potentially new landscape structures and that these can and should be studied independently. This is not a claim that the family of landscapes used by an algorithm should not be studied as a whole, just that it is possible to study the components in isolation, that this will be a simpler task and that it is worthwhile.

The fact that the model does not *require* a distance metric is not a limitation. There is nothing in the model that prevents the definition of a distance metric on $\mathcal{R}$ or V, and if this proves useful, it should be taken advantage of. In cases where the landscape is fully connected, there is always a natural definition of distance, and this can be used to compute such things as correlation lengths [65]. In other cases there may be no useful definition of distance. That the model does not provide one is not a shortcoming. It cannot be denied that the algorithm is making moves on the graph defined by the operator. A landscape model that provides a concrete, well-defined graph which can be studied, is far better than nothing.

2.11. Operators and Representation

The aim of this section is to make it clear that representation and operators are not independent and to illustrate how they interact. The choice of one constrains the possible choices of the other. To some this will be obvious, but others (I was one) may not have realized how tightly bound the two are. A choice of $\mathcal{D}$ is also a choice to use operators which act on that data structure.

There is an important tradeoff in these choices. Typically, the more intuition that has gone into the choice of $\mathcal{O}$, the more difficult it will be to design operators that act on and produce members of $\mathcal{O}$. This is related to the problems with maintaining legal objects which are addressed in §2.12(41). Here I am more concerned with the tradeoff between the recognition of important classes of objects and the construction of operators to move between them.

An example that will be relevant to experiments of this dissertation concerns the representation of Turing machines. A simple representation will ignore the fact that there are many symmetries amongst Turing machines that, if removed, would result in far smaller search spaces. The simple solution is to ignore these symmetries, which allows the use of simple operators. Once the symmetries are noted, they can be excluded from consideration by adopting some canonical Turing machine representation, but it may be difficult to construct operators that produce canonical machines from other canonical machines. This particular example is dealt with in greater detail in §3.8.5.1(61). Unfortunately, reducing the size of the space that is being searched does not necessarily make the search problem simpler (see, for example, [66, 67]). The tradeoff between representations and operators will be encountered again in §5.3.1(124).

2.12. Objects and Representation

There is much that can be said about the choices of $\mathcal{O}$ and $\mathcal{R}$ and the relation, Γ, between them. As mentioned in §2.3(20) a choice of representation is a choice of data structure and this choice leads immediately to a set $\mathcal{R}$ whose members correspond to the various ways the bits in a data structure representing an object can be filled. Note that this does *not* restrict attention to fixed length data structures.

The problem of course is that there may be many ways of filling a data structure that do not result in the representation of an object in $\mathcal{O}$. A similar problem exists when the problem statement involves constraints on the form of the solution. The representation space that is chosen may include elements that correspond to objects that violate constraints. These situations are very common in evolutionary algorithms, and a number of ways of dealing with them have been adopted. They arise in even the simplest GA applications (for example, using 2 bits to represent 3 objects), in algorithms that manipulate permutations (where crossover can easily produce a non-permutation), and with floating point representations. There have been several approaches to dealing with these problems:

- Probably the most common solution is to build special operators that produce legal representations from other legal representations. This is a common approach when manipulating permutations of integers. Operators for this include Cycle Crossover [58], Order Crossover [58], Partially Matched Crossover [68], Edge Recombination [59, 69, 70], the crossover of Gorges-Schleuter [71], Maximal Preservative Crossover [72], Strategic Edge Recombination [73], and Generalized N-point Crossover [67]. Michalewicz describes special-purpose operators designed to stay within a feasible region of $\mathbb{R}^n$ given by linear constraints [74].

- Another solution is to allow these illegal representations but to penalize them somehow to encourage the algorithm to avoid these regions of the space, possibly with increasing probability over time [5, 54, 75, 76, 77, 78, 79].

- Davis and Steenstrup [80] suggest that the problem can be dealt with by "decoding" illegal individuals before evaluating them. They do not give an example, but claim the procedure is often computationally intensive.

- Another approach is to allow the operators to construct illegal representations but to repair the result [81, 82], a procedure called "forcing" by Nakano et al. [83].

- If the size of the subset of $\mathcal{R}$ that is legal is not too small, a solution is to generate individuals repeatedly until a legal one is found.

- A recent elegant solution, proposed by Bean, adopts a new representation of a permutation that allows traditional operators to be used [84].

This dissertation does not attempt to treat representational issues such as the above, though this aspect of search is as important as any other (see §2.13(42)).

2.13. Origins and Choices

This section discusses the origins of some of the components of search algorithms and landscapes. The importance of the choice of $\mathcal{O}$, the objects that are being considered as possible solutions to the search, was illustrated in §1.3(5) with the eight queens problem. This is by no means an isolated example. This choice might not seem part of the search algorithm, but it is simple to view it as such. Perhaps the reason why this aspect of search is not so considered is that we typically have no idea how such a choice could be performed by a machine. The choice requires experience, insight and creativity. It is possible for an algorithm to make this choice and to explore various answers during the course of a search, but currently we do not know how the process works well enough to design an algorithm to perform it. If this component of search eventually falls into the domain of the computation, artificial intelligence will have taken a significant step.

The choice of the representation space, $\mathcal{R}$, has similar importance and is also typically made by the programmer, not the program. An important issue is that of what I call *over-representation* and *under-representation*. Over-representation occurs when many elements of $\mathcal{R}$ are interpreted (according to Γ^{-1}) as corresponding to the same element of $\mathcal{O}$. From a problem solving point of view, such a choice is strictly redundant, but it may have advantages that we do not yet fully appreciate [66, 67]. The mapping between RNA primary and secondary structure is highly redundant in this sense. GP has a similar flavor; every S-expression is a member of an infinitely large class of S-expressions that are all functionally equivalent.

Under-representation is very common in problem solving. A dramatic and elegant example is Kanerva's Sparse Distributed Memory in which an extremely large address space (e.g., of 2^{1000} locations) is represented in a conventional-sized memory [41]. As mentioned earlier, an object space that involves an infinite set must be under-represented at any one point in time. Genetic algorithms have traditionally under-represented real intervals via a discretization of the interval indexed by the binary value of a bit string. Using a floating point representation also under-represents the reals, but in a far less drastic manner. Work on issues of representation, particularly changing representation, can be found in [47, 48, 49, 50, 51, 85].

A representational choice that is common in AI and OR has the elements of $\mathcal{R}$ represent *partial* objects from $\mathcal{O}$. A partial object can also be viewed as representing the class of objects that in some sense contain the partial object. For example, a sub-tour in a graph can be thought of as a partial tour, but also as a representative of all those complete tours that include the sub-tour. This form of representation is the basis of the "split-and-prune" [31] paradigm of OR and it underlies all of the many variants of the branch-and-bound algorithm. A *schema* in a GA is also representative of a class of objects (in this case binary strings), though the GA does not manipulate schemata explicitly,[3] though each binary string of length n is an instance of 3^n schemata and thus operations on individuals can be thought of as operations on samples from many schemata. This was described as *intrinsic parallelism* by Holland in [9] but has become more widely known as *implicit parallelism*.

The final choice I want to mention is that of operators. Every instance of every search problem has a perfect operator. If the operator could be found (which of course it cannot), very little else would matter. A random walk search strategy would locate a global optimum in at most two evaluations. This operator would build a graph similar to that shown in Figure 6. Paradoxically, an operator that would be even

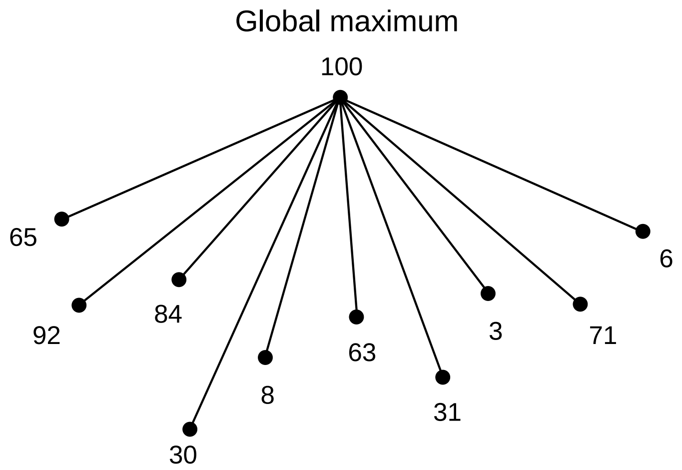

Figure 6. The landscape generated by a perfect operator. Every point in the space, except the global optimum is connected directly to the global optimum. Vertices are labeled with fitness values. Any search strategy using this operator will find the global optimum quite rapidly.

harder to construct would result in a difficult problem. This operator connects each vertex directly to the vertex that has the next highest fitness. This is essentially the landscape that was produced by Horn et al. in [86], though that landscape was produced by combining a very specific fitness function with an ordinary operator and this is produced by combining a very specific operator with an ordinary fitness function. An example is shown in Figure 7. The importance of the choice of fitness function is discussed at length in the second half of Chapter 5.

[3]This is not strictly true if a binary string with no don't care symbols is regarded as a schema, which it is sensible to do. A more accurate statement would assert that schemata of order greater than zero are not explicitly manipulated by the GA.

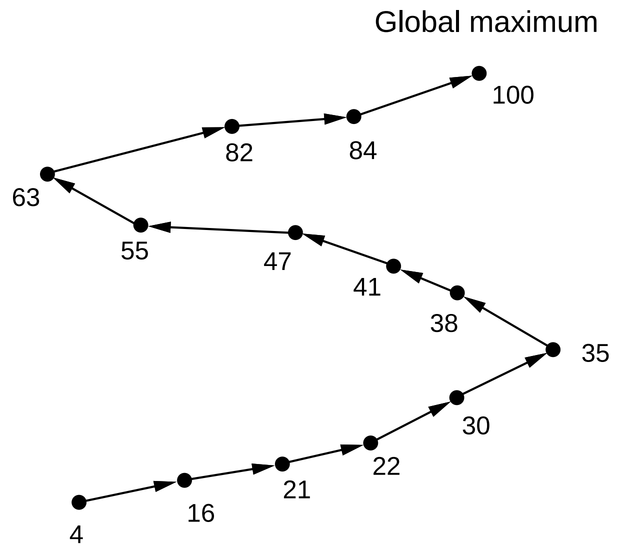

Figure 7. The landscape generated by a not-so-perfect operator. Every point in the space, except the global optimum is connected directly to point with next highest fitness. Vertices are labeled with possible fitness values. Any search strategy that starts at a randomly chosen vertex and thereafter uses this operator alone to generate new possible solutions will explore an average of half the space on each successful search.

The point of these remarks and illustrations is simply to emphasize that in all cases these choices have to be made and that the choices that are made may have a tremendous impact on the difficulty of a problem. I am not suggesting that, given a search problem, we should search for the perfect operator or fitness function. In every case, the choices that are made determine a landscape as described by the current model. This suggests instead that we should try to develop techniques to study these structures, and attempt to apply what we learn to the problem of deciding whether one choice in their construction appears better than another. The landscape model that underlies these structures is in all cases a graph, so a technique developed to study an aspect of one landscape can be automatically applied to many others.

2.14. The Usefulness of the Landscape Metaphor

The term "landscape" has something powerfully seductive about it. The imagery it evokes is so appealing, that further thought can be completely suspended. An important question to ask is why we would want to use such a term. The answer is presumably that we hope to use the imagery (e.g., peaks, ridges, valleys etc.) to enhance our understanding of some process, to develop new ideas for exploring spaces and to stimulate questions about processes operating on these structures. All of this tends to rely rather heavily on the simple properties that we see in physical three dimensional landscapes. It is not clear just how many of the ideas scale up to landscapes with tens or thousands of dimensions. It is quite possible that the simplicity and beauty of the metaphor is actually damaging in some instances, for example by diverting attention from the actual process or by suggesting appealing, simple and incorrect explanations. Many of these potential problems are summarized by Provine [87, pp. 307–317], which should be required reading for people interested in employing the metaphor. Wright's response to Provine's criticism is that his landscape diagrams were intended as a convenient, but simplistic, representation of a complex process in a high-dimensional space [37], and were never intended for mathematical use.

As outlined in §2.1, similar problems exist in evolutionary computation. Given this, it is worth asking whether it is better to abandon the term or to use it and try to be more precise about what is actually meant. There is something to be said for abandoning it. After all, in virtually every formulation, a landscape can be regarded as a graph. On the other hand, it seems unlikely that the term will just go away. In addition, the metaphor, however distant it may sometimes be from reality, *has* given rise to new ideas and intuitions. I have chosen to adopt the term, with the hope that it will lessen, rather than increase, the vagueness with which it is applied.

2.15. Conclusion

This chapter presented a general model of landscapes and an overview of its consequences, advantages, limitations and relevance to evolutionary algorithms. The model views a landscape as a directed graph whose edges and vertices are labeled. It was argued that the operators in evolutionary algorithms each generate a landscape, that these landscapes have differing qualities, and that each can and should be studied in its own right. Thus most evolutionary algorithms are seen as operating on multiple landscapes. Defining a landscape as a graph establishes a contact with search algorithms from artificial intelligence and operations research, many of which are explicitly designed to search labeled graphs. The model advocates a view of search as composed of navigation and structure, with the structure provided by landscapes. It is argued that the statistical properties of landscapes can be studied independently of navigation strategies. The relationship between this model and other work on landscapes is dealt with in detail in Chapter 6. Most of the issues touched on in this chapter will be encountered in the chapters that follow.