CHAPTER 4

Reverse Hillclimbing

4.1. Introduction

There are many algorithms that can be informally described as hillclimbers. Though simplistic, many hillclimbers prove surprisingly efficient in some settings. Slight enhancements can result in algorithms which are amongst the most useful (e.g., simulated annealing). Despite the simplicity of hillclimbing algorithms, it is difficult to accurately compare these algorithms to each other. To compare the performance of two hillclimbers, the simplest method is to run them both a large number of times. The results give a statistical indication of which algorithm is superior. This chapter introduces a new technique, *reverse hillclimbing*, which, in some cases, allows a much more direct comparison. This technique is used to compare four hillclimbing algorithms on three instances of two problems.

The method also makes it possible to compute statistics regarding basins of attraction in a landscape graph. Given a peak on a landscape, reverse hillclimbing computes the exact size of the basin of the point under a hillclimbing algorithm and the exact probability that the peak will be found via a single hillclimb. An attempt to obtain similar statistics using hillclimbing to form estimates would probably consume inordinate amounts of time. For example, a typical global optimum in the 4-state busy beaver problem has a basin of attraction under steepest ascent hillclimbing of size approximately 5000. The probability of locating such a peak from a randomly started hillclimb is usually about 1 in 15 million. Thus one would expect to perform 15 million hillclimbs before stumbling on the peak even once. This would net a few of the 5000 vertices in the basin. Obtaining a suitably reliable estimate (let alone the exact number) of the basin's size would require a far greater number of hillclimbs. Reverse hillclimbing can provide exact answers to these questions, and others, in under a minute with very limited computational requirements. To my knowledge, none of the information provided by reverse hillclimbing could be obtained exactly in the past without an exhaustive enumeration of the vertices in the landscape. Perhaps as a result, there are few results that deal with basin sizes.

Given a vertex v in the graph of a landscape $\mathcal{L} = (\mathcal{R}, \phi, f, \mathcal{F}, >_{\mathcal{F}})$, reverse hillclimbing provides exact answers to the following questions about the basin of attraction of v, $B_\phi(v)$:

- What are the elements of $B_\phi(v)$?
- By how many paths can each element of $B_\phi(v)$ ascend to v? What are the lengths of these paths and what is the expected number of evaluations along each path?
- What is the probability that a single hillclimb, started from a randomly chosen vertex, will ascend to v?
- How many elements of $B_\phi(v)$ can ascend only to v?

When reverse hillclimbing is applied to a large number of peaks on a landscape, the collection of peak fitnesses, basin sizes and ascent probabilities, allows the examination of the relationships between

- peak height and basin size,
- peak height and the probability of peak discovery, and
- basin size and probability of peak discovery.

4.2. Local Search Algorithms

Given the graph $G_{\mathcal{L}} = (V, E)$ of a landscape $\mathcal{L} = (\mathcal{R}, \phi, f, \mathcal{F}, >_{\mathcal{F}})$, *Local search algorithms* are those algorithms whose behavior is shown in Algorithm 4.1. This defines a broad class of algorithms that can be thought of as taking a random walk using their operator. The fitnesses encountered in such a walk will depend entirely on the operator used. If $\mathcal{R} = \{0, 1\}^n$ and β, the bit-flipping operator described in §2.6.1(31), is used, then any point can be reached from any other. Local search algorithms that are more useful than random search on a given class of problems will use operators that tend to produce higher fitness points from lower.

```
choose a vertex v ∈ V;
while (not done) {
      Use φ to generate w ∈ N_φ(v);
      v ← w;
}
```

Algorithm 4.1 Local search algorithm.

The initial vertex is commonly chosen by uniform random selection from the representation space. There are many other possibilities. For example, the search may commence from a vertex generated by another search method, or the initial vertex may be chosen according to some belief about the nature of the landscape, or may be a starting configuration that is given by the statement of the problem. The second step, generating $w \in N_\phi(v)$ also has many variants. In the algorithms which we will consider, ϕ will employ β to generate $N \subseteq N_\beta(v)$, one element of which, w, will be selected. As examples, some operators will use β to generate $N = N_\phi(v)$, the entire β-neighborhood of v and then select the fittest. Others will use β to generate a fixed number of β-neighbors of v, or generate β-neighbors until a vertex fitter than v is located or some limit is reached. In the final step, the algorithm either terminates or loops for another iteration. The decision of when to terminate a search may be according to some maximum number of evaluations, the achievement of a desired fitness level, the detection of a local maximum, the result of running out of time for the search, or some other method. The above description also deliberately makes no mention of the return value of the algorithm. Typically these algorithms keep track of the best element of $\mathcal{R}$ encountered during the search and return that. Algorithms that never select a w such that $f(w) <_{\mathcal{F}} f(v)$ can simply return v when they are done.

4.2.1. Iterated Local Search Algorithms

A simple enhancement to a local search algorithm is an iterated version which repeats the original local search algorithm some number of times. These algorithms are often employed on problems in which the initial vertex is chosen uniformly at random, though there are important exceptions [116].

4.3. Hillclimbing Algorithms

The definition of local search algorithms provides a framework for the definition of hillclimbing algorithms. A hillclimbing algorithm is a local search algorithm that always selects a w in step 2 above such that $f(w) \geq_{\mathcal{F}} f(v)$. Under this definition, "hillclimbing" means that the algorithm never goes downhill. One argument for including algorithms that sometimes select a w such that $f(w) = f(v)$ is that every algorithm which makes strictly increasing moves can be simply modified to also allow moves to vertices of equal fitness. Also, this definition of a hillclimber is in accordance with Minsky's early description of hillclimbers as climbing hills (obviously), but also as getting trapped at "false peaks" and as potentially wandering on "mesas" of equal fitness [25].

Dividing local search algorithms on the basis of the relative fitnesses of v and w leads to three classes, $LSA_{<=>}$, $LSA_{>=}$ and $LSA_{>}$, with $LSA_{>} \subset LSA_{>=} \subset LSA_{<=>}$. These are depicted in Figure 44. Hillclimbers are those algorithms found in the two inner circles. This taxonomy will allow easy discussion of a number of algorithms and delineation of the class algorithms upon which reverse hillclimbing can be applied. This chapter explores the characteristics of four hillclimbing algorithms from $LSA_{>}$ on a number of problems. These are, Any Ascent, Least Ascent, Median Ascent and Steepest Ascent. Figures 45 to 48 shows the landscape corresponding to each of the operators of these algorithms. In these landscape graphs, $\mathcal{R} = \{0,1\}^n$, f, $\mathcal{F}$, and $>_{\mathcal{F}}$ are held constant.

4.3.1. Any Ascent Hillclimbing

The *any ascent* hillclimbing algorithm (AA), employs an operator that uses β to generate β-neighbors of v until it finds a w such that $f(w) >_{\mathcal{F}} f(v)$. Of course if such a β-neighbor does not exist, the algorithm should not loop indefinitely. This can be avoided by keeping track of the neighbors generated by β, or by using some limit on the number of β-neighbors that can be tested. The first solution, the one used in this chapter, also makes it possible to avoid evaluating β-neighbors multiple times. The second solution results in an algorithm whose sibling in $LSA_{>=}$ is the BH algorithm of Chapter 3.

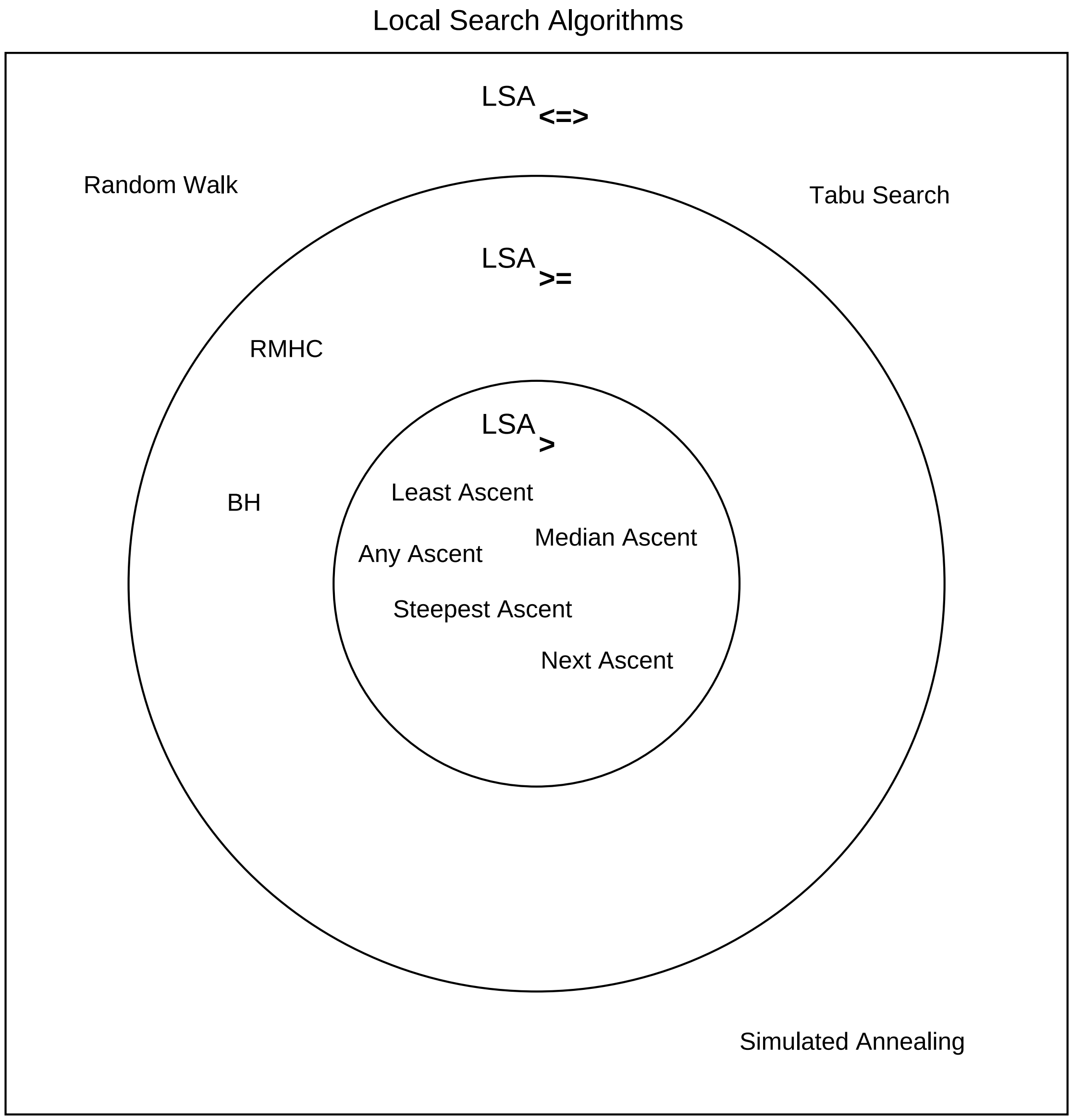


Figure 44. A division of local search algorithms according to the types of fitness moves they make. Hillclimbing algorithms comprise the inner two circles. The RMHC and BH algorithms are described in Chapter 3.

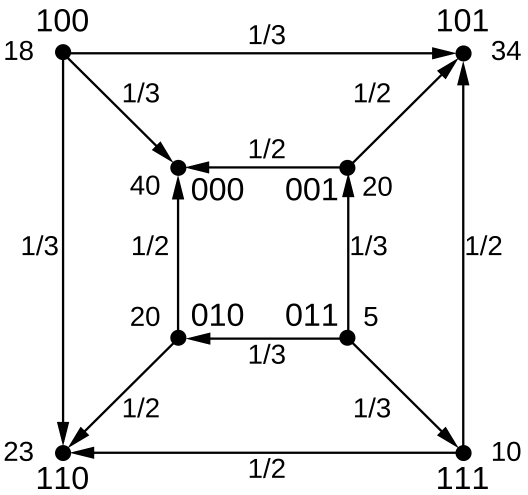


Figure 45. The landscape produced by the *any ascent* operator given a representation space $\{0,1\}^3$ and a hypothetical fitness function. Vertices are labeled with fitnesses and edges with transition probabilities.

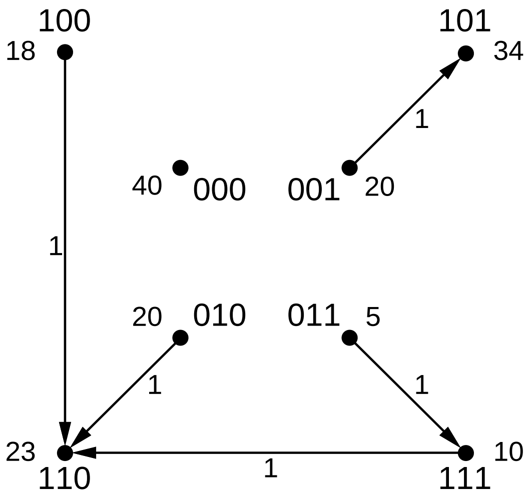


Figure 46. The landscape produced by the *least ascent* operator given a representation space $\{0,1\}^3$ and a hypothetical fitness function. Vertices are labeled with fitnesses and edges with transition probabilities.

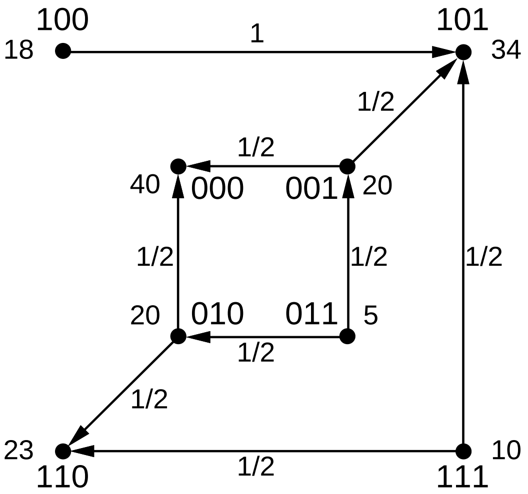


Figure 47. The landscape produced by the *median ascent* operator given a representation space $\{0,1\}^3$ and a hypothetical fitness function. Vertices are labeled with fitnesses and edges with transition probabilities.

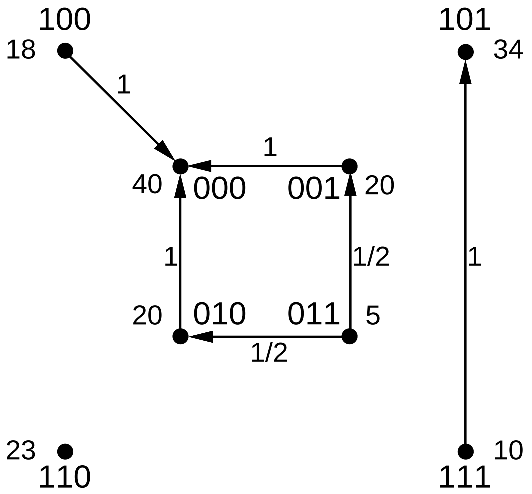


Figure 48. The landscape produced by the *steepest ascent* operator given a representation space $\{0,1\}^3$ and a hypothetical fitness function. Vertices are labeled with fitnesses and edges with transition probabilities.

4.3.2. Least Ascent Hillclimbing

The *least ascent* hillclimbing algorithm (LA), employs an operator which uses β to generate all of $N_\beta(v)$ and sets w to an element of $N_\beta(v)$ with least fitness subject to the condition $f(w) \geq_{\mathcal{F}} f(v)$. In other words, from the subset of $N_\beta(v)$ with fitness greater than v the least fit element is selected. This algorithm therefore takes a conservative approach to hillclimbing, never moving uphill more than necessary. There may be several equally fit elements of $N_\beta(v)$ that could be selected. In this case, one is chosen uniformly at random.

4.3.3. Median Ascent Hillclimbing

The *median ascent* hillclimbing algorithm (MA), employs an operator which uses β to generate all of $N_\beta(v)$ and sets w to an element of $N_\beta(v)$ with median fitness amongst those with fitness greater than $f(v)$. This algorithm therefore takes a middle-of-the-road approach to hillclimbing, moving to the middle of all the uphill neighbors. There may be an even number of fitter β-neighbors. In this case, one of the two medians is chosen uniformly at random. Vertex 011 in Figure 47 shows an example of this.

4.3.4. Steepest Ascent Hillclimbing

The *steepest ascent* hillclimbing algorithm (SA), employs an operator which uses β to generate all of $N_\beta(v)$ and sets w to an element of $N_\beta(v)$ with greatest fitness. There may be several maximally fit elements of $N_\beta(v)$ that could be selected. In this case, one is chosen uniformly at random. Vertex 011 in Figure 48 shows an example of this.

4.4. The Reverse Hillclimbing Algorithm

This section describes the basic reverse hillclimbing algorithm, shows how it may be enhanced to obtain more information, and discusses some implementation issues.

4.4.1. The Basic Algorithm

The reverse hillclimbing algorithm is conceptually quite simple. Algorithm 4.2 shows a recursive function that collects the vertices in the basin of attraction of a vertex v into a set S (initialized to the empty set) given a hillclimbing algorithm, $H \in LSA_>$, that uses an operator ϕ. When the recursion terminates, the set S will contain the basin of attraction of v for operator ϕ. The above function is not an efficient implementation

```
basin_size(v)
VERTEX v;
{
    S = S ∪ {v};

    for each (w|v ∈ N_φ(w) and f(w) <_F f(v))
        basin_size(w);
}
```

Algorithm 4.2 The basic reverse hillclimbing algorithm. S, an external variable, is initialized to the empty set before the initial invocation of the function. S will contain the vertices in the basin of attraction when the recursion terminates.

for many reasons Some of these are discussed in §4.4.2 and Appendix A. There are several points that should be noted about this algorithm:

- Given a representation, $\mathcal{R}$, and fitness function, the basin of attraction of an element of $\mathcal{R}$ depends on the operator in use. This can be seen by considering the basins of attraction for the vertex 110 in Figures 45 to 48.

- Assuming the set of vertices that may have v as a neighbor can be identified without search, reverse hillclimbing can be implemented so that it never examines a vertex in the space that is more than a single step from a vertex in the basin of attraction. When vertices outside the basin of attraction are examined, they do not become the subject of a subsequent recursive call. This leads immediately to an upper bound on the number of vertices examined by the algorithm of $n|B_\phi(v)|$ where n is the maximum in-degree in the landscape and $|B_\phi(v)|$ is the size of the basin of attraction. In practice, the number of elements of $\mathcal{R}$ evaluated will be far smaller than this for two reasons: first, it is common for several less fit neighbors of a vertex to be able to re-ascend to that vertex (for instance in AA they all can) and second, the sets of downhill neighbors of two vertices will often not be disjoint.

- The function *basin_size* may be called more than once on the same vertex. To see why this is true, consider Figure 49 which shows two paths to the same vertex in a one max problem.

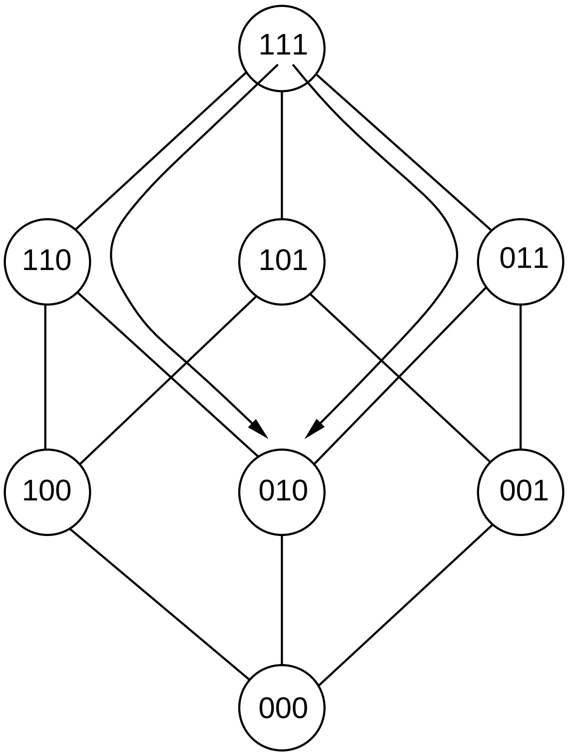


Figure 49. Two downhill trajectories when performing reverse hillclimbing on a one max problem with strings of length three. The arrows illustrate how a vertex (in this case 010) can be encountered more than once. In the basic algorithm, this vertex would be the subject of two recursive calls to the *basin_size* function.

4.4.2. Augmenting the Basic Algorithm

The basic reverse hillclimbing algorithm given above only produces the vertices in the basin of attraction of a vertex. This algorithm may be augmented to provide the additional details of the basin and the vertices in it that were mentioned in the introduction: ascent probabilities, number and lengths of ascending paths, and expected evaluations to peak discovery.

Because the algorithm will typically encounter vertices in the basin on more than one occasion, it is necessary to store partial information about vertices until they have been encountered for the final time, at which point they can be output or their properties can be added to accumulating statistics about the basin. A straightforward solution to this problem is to use a hash table to implement the set S above to keep track of the details of vertices that have been encountered. This allows the maintenance of details of probability of ascent, number of ways to ascend and expected comparisons to ascend for each vertex in the basin. When the function *basin_size* discovers a vertex that could ascend to the current argument, it checks the hash table to see if this vertex has already been encountered. If so, it updates the data structure containing the information for the vertex. If not, it creates a new data structure and inserts it into the hash table. A rough outline of how each statistic is kept follows:

- If a vertex v_1 can ascend along some path to the original vertex v with probability p, and it is discovered that v_2, a less fit neighbor of v_1, can ascend to v_1 with probability q, then pq is added to the overall probability that v_2 can ascend to v. Note that the algorithm may encounter v_1 again (by finding another downhill re-ascending path to it), and this will cause the probabilities associated with both v_1 and v_2 to be incremented again. When the recursion is finished, every downhill path will have been taken, and the probabilities associated with the vertices in the basins will be correct.

- It is easy to keep track of the number of paths by which a vertex may ascend to the original vertex. Simply keep a count of the number of times each vertex in the basin is encountered during the recursion (including the first, in which the vertex is not found in the hash table).

- If a vertex can only ascend to a single peak under a given hillclimbing algorithm, the vertex will be said to be *owned* by the peak that is reached. To calculate how many of the vertices in the basin of attraction are owned by the peak, look at the final ascent probabilities. Those vertices with probability one are owned.

- Calculating the expected number of vertex fitness evaluations it will take a vertex to reach the peak is similar to the calculation of ascent probability. Information is passed down through the basin as paths are traversed. Three of the four hillclimbers, LA, MA and SA, use β to produce all the β-neighbors, so their expected evaluations per uphill step is simply the number of β-neighbors at each step. To calculate the expected number of evaluations for an uphill step of AA, it is necessary to determine the number of uphill β-neighbors that exist at that vertex. If we let n be the number of β-neighbors and u be the number of uphill β-neighbors, then the expected number of evaluations before the vertex is left is given by

$$\begin{aligned} E(evaluations) &= \sum_{i=1}^{\infty} i \left(\frac{n-u}{n}\right)^{i-1} \frac{u}{n} \\ &= \frac{u}{n} \frac{1}{\left(1 - \frac{n-u}{n}\right)^2} \\ &= \frac{u}{n} \frac{1}{\left(\frac{u}{n}\right)^2} \\ &= \frac{n}{u} \end{aligned}$$

The second line of this derivation follows from the equation obtained by differentiating the identity $\frac{1}{1-x} = 1 + x + x^2 + x^3 + \cdots$.

This list is merely a brief overview of how further details of the basin of attraction and its vertices may be obtained. It should provide enough information for the algorithm to be implemented. Further discussion of implementation is provided in Appendix A.

4.5. An Important Tradeoff

There is an important tradeoff between hillclimbing and reverse hillclimbing. These algorithms can both be used to provide useful information about a landscape. Typically what one algorithm can do well, the other cannot. For example, on a large landscape with a single peak whose basin of attraction is the entire landscape, it is impractical to calculate the size of the basin using reverse hillclimbing. The basin may have more vertices than there are atoms in the universe, in which case it would would be far more efficient to arrive at this conclusion, with some (estimable) probability of being wrong, by undertaking a million hillclimbs from random vertices and observing that they all arrived at the same place. If, however, a peak has been found through some means and one wishes to calculate the probability of this occurring (for instance to test an hypothesis that the algorithm that found the peak was merely lucky), using hillclimbing to form an estimate of the peak's basin size and probability of attraction can be equally time-consuming. Reverse hillclimbing may be able to provide the answer rapidly.

The reverse hillclimbing algorithm has the pleasing property of becoming more effective as the landscapes on which it is set to work become more difficult to search.

As problems get harder and bigger, it may be that reverse hillclimbing can *only* practically be employed to find details of the hardest to locate areas of these hardest problems. Naturally, the algorithm does not help to identify these areas before setting to work on exploring them.

4.6. When can a Hillclimber be Reversed?

I have defined hillclimbers to be the algorithms in $LSA_{>=}$, but this chapter deals with four hillclimbers that are members of $LSA_{>}$, a proper subset of $LSA_{>=}$. This raises some questions:

- **Can reverse hillclimbing be applied to all algorithms in $LSA_{>}$?** That is, can it be applied to all hillclimbing algorithms that only take uphill steps? I believe the answer to this question is that it can in theory. In practice, there are three reasons why this may be difficult or impractical:

    1. The neighborhood of vertices in the landscape may be too large to completely examine. In the extreme, the neighborhood of each vertex includes every other vertex. This is the case with the mutation operators of GAs. It is not practical to perform reverse hillclimbing on this landscape for exactly the same reason that it is not practical to perform SA hillclimbing on it.
    2. The calculation of ascent probabilities might be difficult for some algorithms.
    3. The hillclimbing algorithm may be relatively difficult to reverse, as in the case of *next ascent*. This does not prevent reverse hillclimbing from being done, but will make its implementation more challenging.

- **Can reverse hillclimbing be applied to algorithms in $LSA_{>=} - LSA_{>}$?** That is, can one perform reverse hillclimbing on a hillclimbing algorithm that potentially makes moves to vertices of equal fitness? The simplest answer is that this appears difficult, since these algorithms may spend an arbitrary amount of time exploring a plateau. If the reverse hillclimbing algorithm was provided with analytically derived results to deal with the expected results of this behavior, the situation might be improved, but even that seems doubtful. The amount of information that would need to be provided to the reverse hillclimbing algorithm so it could avoid infinite loops would likely be enough to solve the original problem. These algorithms might be analyzed completely using techniques such as Markov chains, though in practice there will be far too many states in such a Markov process to make study of even moderate landscapes feasible.

All is not lost however, as it is still possible to determine the basin of attraction of a peak. The hash table-based reverse hillclimbing algorithm can be modified to include vertices of equal fitness in the basin, but not attempt to calculate the number of ascent paths or the probabilities of these paths.

- **Can reverse hillclimbing be applied to algorithms in $LSA_{<=>} - LSA_{>=}$?** That is, can the method be applied to local search algorithms that potentially take downhill steps? The answer appears to be no. As above, it may still be possible to determine the basin of attraction (without knowing the number of paths, their probabilities and their expected evaluations etc.), but to get the full complement of information, it appears that other methods would need to be used, and it is not clear what these might be.

4.7. Test Problems

This chapter demonstrates how reverse hillclimbing can be used to compare hillclimbing algorithms via examination of the basins of attraction that arise from instances of two search problems. The first is the busy beaver problem, and the second is the problem of locating high fitness vertices on NK landscapes.

4.7.1. NK Landscapes

The NK landscapes were introduced by Kauffman [117, 118] as a simplistic model of systems with many interacting parts, particularly the complex epistatic interactions found in genomic systems. An NK landscape has two important parameters: N and K. N is the number of dimensions in the landscape, each dimension having some fixed number of possible values. Attention has been focused almost exclusively on binary dimensions, and this is also the approach of this chapter. A vertex in an NK landscape can therefore be represented by a binary string of length N. The N positions are often referred to as loci.

The fitness of a vertex in an NK landscape is computed in an additive fashion. Each of the N loci makes a contribution to the overall fitness. These are then summed and divided by N to obtain the fitness of the vertex. The variable K controls the degree to which the fitness contribution of each locus is affected by the binary values found at other loci. K is the average number of other loci that affect the fitness contribution of each locus. If $K = 0$, the fitness contribution of each locus is independent of every other locus, and the optimal value for each locus can be determined by simply comparing the overall fitness of two points—one with the locus set to 0 and the other with the locus set to 1. If $K = N - 1$, every locus is affected by every other locus and changing the value assigned to a locus also changes the fitness contribution of every other locus.

A simplistic way to implement an NK landscape requires filling an N by 2^{K+1} array with real numbers chosen uniformly at random from the interval $[0{\cdot}0, 1{\cdot}0]$. A simple method of choosing influencing loci is to consider the N loci to form a ring, and let the $\lfloor K/2 \rfloor$ loci to the left and $\lceil K/2 \rceil$ loci to the right of each locus influence that locus. An NK landscape with $N = 4$ and $K = 2$ and an example fitness calculation is shown in Figure 50. Neighborhood in an NK landscape is defined by the operator β defined in §2.6.1(31). In general, one cannot store $N2^{K+1}$ random numbers and so this simplistic method must be replaced with something more sophisticated, and there are several ways in which this can be done. This presentation of NK landscapes is deliberately very brief, as more detailed presentations are easily available [117, 118].

If N is fixed and the value of K is allowed to vary, the set of landscapes that are produced will range from almost certainly unimodal (containing a single global maximum) for $K = 0$, to completely random (meaning that the fitness of a vertex is completely uncorrelated with that of any of its N one-mutant neighbors) for $K = N - 1$. The NK landscapes are thus said to be *tunably rugged*. Because there are aspects of some NK landscapes that are amenable to analytical investigation and because there are other aspects of these landscapes that have been studied empirically [117, 118], these landscapes provide a good testbed for reverse hillclimbing studies.

4.7.2. Busy Beavers

The busy beaver problem is described in §3.8.5(60).

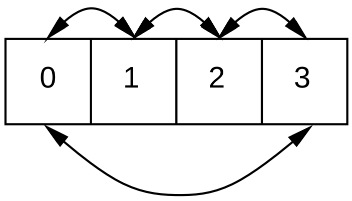
4.8. Experiment Overview

Reverse hillclimbing makes it possible to study many things. These include details about particular landscapes or ensembles of landscapes, properties of hillclimbing algorithms and relationships between hillclimbing algorithms. The experiments of this chapter investigate the relative performance of the various hillclimbing algorithms on a set of peaks. In these experiments, the aim is to provide an answer to the

Locus

Neighborhood	0	1	2	3
000	.18	.04	.73	.91
001	.42	.77	.35	.11
010	.68	.31	.04	.89
011	.91	.17	.25	.70
100	.93	.12	.73	.53
101	.59	.64	.82	.77
110	.19	.94	.38	.21
111	.43	.49	.62	.55

Influencers



Fitness calculation for string 0110

Locus	Neighborhood	Fitness
0	001	.42
1	011	.17
2	110	.38
3	100	.53

Fitness = (.42 + .17 + .38 + .53) / 4

= .375

Figure 50. The components of a simple NK landscape where $N = 4$ and $K = 2$. The table on the left shows the fitness contribution (rounded to two decimal places) for each locus for each possible neighborhood configuration. The small diagram at the top right shows the loci that influence each locus. An illustration of the fitness calculation for the vertex 0110 is shown in the bottom right.

question: Which hillclimbing algorithm can be expected to locate a peak using the fewest function evaluations? The experiment considers the randomly located peaks and also sets of high-fitness peaks. On the studied instances of the busy beaver problem, the optimal peaks are all known, and the performance of the hillclimbers is compared given the task of locating any one of the optimal peaks. For the NK landscapes, the best five percent of the randomly located peaks are separated, and the hillclimbers are compared when their task is to find any one of these peaks. The reverse hillclimbing algorithm is used to make predictions about the performance of the four hillclimbers and experiments show these to be very accurate.

The reverse hillclimbing analysis helps to throw light on an important factor in hillclimbing algorithms. This is the question of how much time should be spent looking for uphill neighbors before moving. Reverse hillclimbing makes it possible to see that, on the problems considered, hillclimbers that choose higher neighbors have higher probabilities of reaching higher peaks. Choosing a higher neighbor incurs a cost however. The hillclimber must examine more neighbors. Finding a good balance between these two is important to overall performance. AA and SA represent quite different responses to this tradeoff. AA performs few function evaluations to find any uphill neighbor whereas SA examines all neighbors to guarantee that one with maximal fitness is selected. Several new hillclimbers that explore this tradeoff are examined in Appendix B.

In the experiments of this chapter, three NK landscapes were investigated. In each of these, the value of N was 16. K took on values of 4, 8 and 12. The 2-, 3- and 4-state busy beaver problems are examined.

4.9. Results

4.9.1. NK Landscapes

For each of the three NK landscapes examined, a number of peaks were identified (via hillclimbing). These were then subject to reverse hillclimbing using each of the four hillclimbing algorithms. Then, for each of these landscapes, the fittest five percent of the peaks were examined separately to see if one of the algorithms appeared better at finding higher peaks.

Table 7 shows the result of reverse hillclimbs from 1000 peaks on an $N = 16$, $K = 12$ landscape. The peaks were located by performing hillclimbs from randomly chosen starting points. The four columns of the table represent the four algorithms. The first row of data shows the number of different basin sizes that were found. For example, of the 1000 different peaks whose basins were explored for LA, there were only 176 different basin sizes found. The second, third, fourth and fifth lines show the minimum, maximum, mean and standard deviation of the basin sizes that were found. The second last line shows the overall probability that a hillclimb started from a randomly chosen location will reach one of the peaks from the set of 1000. The final row is the expected number of climbs that it would take for this to occur. This row is the reciprocal of the previous row.

Table 7. Reverse hillclimbing from 1000 random peaks in a 16,12 NK landscape.

	AA	LA	MA	SA
# of basin sizes	962	176	413	150
Min. basin size	7268	1	1	2
Max. basin size	35,841	541	1230	287
Mean basin size	25,634	35	178	47
Basin size s.d.	4584	62	154	38
P(Discovery)	0·645	0·534	0·619	0·713
E(Climbs)	1·548	1·873	1·617	1·402

There are several things that can be noted about this table:

- AA has basins that are far greater in size than those of any other algorithm. The smallest basin found under AA is approximately six times bigger than the maximum basin size under any other algorithm. That basins under AA are bigger than the basins under the other algorithms is not surprising. If one orders the hillclimbers in terms of decreasing out-degree of landscape vertices, the ranking would be AA first, MA second and LA and SA roughly equal last. The same ranking can be observed in mean (and maximum) basin size. Note that the ranking of algorithms by landscape out-degree would change if N were odd instead of even, since in that case MA would find a single median instead of having to make a random choice between two at every step of its climb.

- AA also produces far more unique basin sizes, which is not surprising given the far greater range of these numbers. Of the 1000 basins that were explored, there were 962 different basin sizes found. What seems initially more remarkable is the low numbers for the other algorithms, especially for LA and SA. This proves to have an uninteresting explanation however, which is that there are many small basins for each of these algorithms. If a hillclimber tends to have small basins, there will tend to be fewer different basin sizes in a sample. This is confirmed by looking at the actual basin sizes (not shown). Further evidence that this is the cause is seen by noting the agreement between the maximum basin sizes and the number of different basin sizes. If we rank the four algorithms by decreasing maximum basin size, and also by decreasing number of basin sizes, we get identical orderings.

- The 1 entries for minimum basin size under LA and MA are eye-catching. If each of the vertices from which reverse hillclimbing is performed in this experiment is supposed to be a peak, how can any of them have a basin size of only one (i.e., the peak and no other vertices)? This seemed at first suspicious but has a simple solution. These vertices are indeed peaks, which is to say that all their neighbors are less fit than they are. But their basins also have size one, because each of the neighbors does not ascend to the peak. This cannot happen under AA since there is always a non-zero probability of an ascent to every fitter neighbor. It is possible in the other three algorithms. It did not happen with SA in this set of peaks, but that will be seen in later experiments.

  These peaks are something of a curiousity, and are without a doubt the hardest peaks to locate with these algorithms, since their basin of attraction contains only the peaks themselves. To find such a peak with a randomly started hill-climb, the peak itself must be chosen as the starting vertex. As a result, LA, MA and SA cannot do better than random search when attempting to locate such a vertex.

- The overall probability of locating a peak shows an interesting pattern. If we ignore AA for the time being, there is a direct correspondence between how exploitative LA, MA and SA are and how often they expect to find one of the set of peaks. In saying that one algorithm is more exploitative than another, it is meant that if all the fitter neighbors of a vertex were discovered and ranked from low to high according to fitness, the more exploitative algorithm would choose to move to one whose rank was higher. SA is more exploitative than MA which in turn is more exploitative than LA. Correspondingly, SA has a higher probability of finding one of the set of peaks than MA which has a higher probability than LA.

Since AA is on average as exploitative as MA (though with far greater variance), it seems reasonable to expect that its probability of discovery lies between that of SA and LA, and closer to MA than to either of the extremes. This is in fact the case. These general patterns of discovery probabilities will persist throughout the experiments in this chapter. At this stage, it should be remembered that these results apply to a set of peaks that were located via randomly started hillclimbs. We have not seen if the pattern persists when we ask what the probabilities are of locating good peaks.

As the 1000 peaks whose basins were calculated were found via randomly started hillclimbs, there is nothing special about them and the fact that one algorithm can find one of them with a higher probability than another is not particularly interesting. More interesting is to consider the best of these peaks to see if any algorithm appears to have an advantage at finding higher peaks. There does not seem to be any a priori reason to expect such a bias, but it is worth looking for. If this bias is not found, it is a good indication that the performance of the hillclimbing algorithms may be accurately estimated via observing their performance on peaks located through random sampling.

Table 8 is identical in form to Table 7, except the figures displayed show only the data for the best five percent (50) of the original 1000 peaks. Points of interest in Table 8 include the following:

- The number of different basin sizes has leveled out across the algorithms. This is to be expected, since we are now sampling the fittest peaks, and it is not unreasonable to expect these to have bigger basins and thus, probably more variance in their basin sizes.

- Interestingly, the 1 entries for LA and MA have persisted, when they might have been expected to disappear as it seems reasonable to expect that they would correspond to particularly poor peaks. This is definitely not the case. Further investigation reveals that 9 of the top 50 peaks have basin sizes of 1 under LA. The neighbors of a very fit vertex, v, are likely to have other neighbors that are less fit than v, and in this case, LA will not move to v from any of its neighbors. MA exhibits the same qualitative behavior, but its effect is greatly reduced (only 1 of the top 50 peaks had a basin of size 1 under MA). Under LA, 15 of the top 50 peaks had a basin of size less than 5, whereas under MA

Table 8. Reverse hillclimbing from the best 50 of 1000 random peaks in a 16,12 NK landscape.

	AA	LA	MA	SA
# of basin sizes	50	30	48	43
Min. basin size	19,242	1	1	28
Max. basin size	35,130	385	1230	287
Mean basin size	29,243	45	313	102
Basin size s.d.	3489	67	241	59
P(Discovery)	0·062	0·034	0·053	0·078
E(Climbs)	16·18	29·39	18·80	12·88

only 2 did. It is not hard to see that the global maximum of these function will always be such a peak under LA and MA! Clearly, LA and MA will not be much help in locating this vertex.

- The pattern of ascent probabilities described above has also persisted. Again, AA comes out slightly ahead of MA.

This information allows us to answer the question of which hillclimber to bet on if we are allowed some number of climbs. But as we are concerned with evaluations, not runs, it would be good to answer the question: Which of the hillclimbing algorithms appears to be the best, in terms of evaluations, at finding this set of peaks? The data from the reverse hillclimbing is exact, but is not enough. If we consider the iterated version of each of these algorithms, a successful search for one of the set of desired peaks will consist of some number of unsuccessful climbs (that do not reach a peak in the desired set) followed by a successful climb. If a problem is at all difficult, there will be many unsuccessful climbs before a peak is found. If a reasonably accurate estimate of the number of evaluations taken by an algorithm on a climb were available, it could be combined with the exact reverse hillclimbing data to obtain an equally accurate prediction of the number of evaluations needed by that algorithm to locate a peak.

Information about the average number of evaluations used in a climb (successful or not) is easily obtained. Table 9 shows a summary of 10,000 climbs on the $N = 16$, $K = 12$ landscape for the four algorithms. As would be expected, AA uses significantly fewer evaluations per climb than the other algorithms, all of which examine all neighbors before moving. The reason LA, MA and SA do not perform similar numbers of evaluations is that they go uphill at different rates, as can be seen from the mean steps per climb in the table. SA has a mean of about 2 steps before reaching a peak, while LA takes about 9.

Table 9. The result of 10,000 hillclimbs on the 16,12 NK landscape. The table shows data for the number of uphill steps taken for each algorithm and the number of evaluations performed per uphill walk.

	AA	LA	MA	SA
Av. steps/climb	3·455	9·335	3·836	1·999
Min. steps/climb	0	0	0	0
Max. steps/climb	13	33	12	8
Steps/climb s.d.	1·878	5·919	1·857	1·064
Av. evals/climb	31·22	166·4	78·37	48·99
Min. evals/climb	17	17	17	17
Max. evals/climb	90	545	209	145
Evals/climb s.d.	10·63	94·70	29·66	17·02

The most interesting question that arises is whether AA will be better than SA or vice-versa. SA has a lower expected number of climbs to locate a peak, but takes more evaluations to do so. The answer to this can be found by multiplying the expected number of climbs from Table 8 by the expected number of evaluations from Table 9. This produces Table 10.

The table compares the expected performance of each of the hillclimbers with their observed performance. SA located more peaks in the 10,000 runs but, more importantly, AA took fewer evaluations on average to locate a peak. Since we are using algorithm evaluation count as our yardstick for algorithm performance, AA is

Table 10. Expected and observed numbers of peaks and evaluations for 50 good NK 16,12 peaks.

	AA	LA	MA	SA
E(Peaks found)	617	340	532	777
Peaks found	620	348	578	736
E(Evals/peak)	505	4889	1473	631
Evals/peak	503	4780	1355	665

the best algorithm for finding this particular set of peaks on this particular landscape, assuming the sampling that gave the expected evaluations per climb data was not wildly wrong. SA is a clear second, with MA and LA third and fourth. These results could easily be made more accurate by increasing the number of hillclimbs done to obtain the data on expected number of evaluations per climb. For our purposes, a small number of hillclimbs to estimate evaluations per climb was sufficient to obtain estimates that closely matched the actual performance of the algorithms.

Of course, the behavior of the different algorithms may simply be due to chance, and may not apply to other NK landscapes, let alone other problems. To investigate this, and also to examine landscapes of differing ruggedness, the same experiment was performed on an $N = 16$, $K = 8$ and an $N = 16$, $K = 4$ NK landscape. Tables 11 to 14 are the $N = 16$, $K = 8$ counterparts of Tables 7 to 10 for the $N = 16$, $K = 12$ landscape. Table 11 shows the result of performing reverse hillclimbing from 700 peaks in a $N = 16$, $K = 8$ landscape. Less peaks are descended from in this experiment as the landscape contains fewer peaks. Judging from the figures for SA in the table, the 700 peaks and their basins account for approximately 96% of the vertices in the landscape. This follows from the fact that SA is with high probability completely deterministic on an NK landscape (since it is very unlikely that any two vertices will have identical fitness), so multiplying the mean basin size (90) by the number of basins (700) gives a good estimate of the number of vertices represented by the sample. In this case the basins represent approximately 63,000 of the 2^{16} vertices. The 2500 or so vertices that are not represented should (if the mean basin size can be used reliably) fall into approximately 28 basins. We would expect only one of these to have fitness in the highest 5% of all basins, so we can be reasonably confident that

Table 11. Reverse hillclimbing from 700 random peaks in a 16,8 NK landscape.

	AA	LA	MA	SA
# of basin sizes	687	233	526	230
Min. basin size	11,179	1	1	1
Max. basin size	49,275	975	4399	973
Mean basin size	34,812	88	583	90
Basin size s.d.	6613	145	659	109
P(Discovery)	0·958	0·945	0·958	0·966
E(Climbs)	1·044	1·059	1·044	1·035

the top 5% of our sample of 700 is very close to being the top 5% of peaks in the entire landscape.

The patterns that were evident in the $N = 16$, $K = 12$ landscape can also be seen on this landscape. Once again, SA has a higher probability of locating a peak than MA does and it in turn has a higher probability than LA. AA has a probability that is very close to that of MA (in the table they are identical as only two decimal places are shown). The basin sizes also show patterns that are the same as those on the more rugged landscape. AA has by far the biggest basins, then MA and then SA just barely ahead of LA. Table 12 shows the reverse hillclimbing results when we restrict attention to the top 5% (35) of the 700 peaks. Again, looking at just the best peaks in the sample does not alter the relative algorithm performances.

Table 13 shows the result of 10,000 hillclimbs on this landscape. Once again it is clear that one of SA and AA will prove to be the best algorithm on this landscape. SA has a better chance of finding a peak on a given climb, but AA takes fewer comparisons per climb. The expected and observed data for the number of peaks discovered and the number of evaluations per discovery is presented in Table 14 and the balance once more favors AA. SA's dominance in probability of discovery does not match AA's dominance in evaluations per climb.

This story is repeated in every detail on the $N = 16$, $K = 4$ landscape. In this case, all 180 peaks in the landscape were identified. The reverse hillclimbs from

Table 12. Reverse hillclimbing from the best 35 of 700 random peaks in a 16,8 NK landscape.

	AA	LA	MA	SA
# of basin sizes	34	31	35	32
Min. basin size	35,751	1	618	73
Max. basin size	48,486	975	14,399	829
Mean basin size	43,704	191	2019	349
Basin size s.d.	3393	236	1044	145
P(Discovery)	0·159	0·102	0·156	0·186
E(Climbs)	6·294	9·809	6·411	5·372

these peaks are shown in Table 15. The best 5% (9) are examined in Table 16, 10,000 hillclimbs were done to gather expected uphill path lengths (Table 17), and the summary results appear in Table 18.

In this case, the basins of attraction of the peaks account for all the vertices in the landscape. As we have seen virtually identical behavior on these three landscapes, it seems reasonable to assume that although we only considered about 70% of the space in the $N = 16$, $K = 12$ landscape, that the results would not have been greatly different had we examined additional peaks.

4.9.2. Busy Beavers

Table 19 shows the result of performing reverse hillclimbing on 1000 randomly located peaks on the 4-state busy beaver landscape. As with the NK landscape experiments above, the peaks are found via hillclimbs started from randomly chosen starting points. With the NK landscapes, the union of the points in all the basins of attraction of the set of randomly chosen peaks comprised a large proportion of the space. For example, with SA, over 70% of the space was in the basin of attraction of the 1000 peaks chosen on the $N = 16$, $K = 12$ landscape (see Table 7 on page 101). This is certainly not the case with the 1000 randomly located peaks in the 4-state busy beaver problem. If we (generously) assume that basins do not overlap, the union of

Table 13. The result of 10,000 hillclimbs on the 16,8 NK landscape. The table shows data for the number of uphill steps taken for each algorithm and the number of evaluations performed per uphill walk.

	AA	LA	MA	SA
Av. steps/climb	4·632	12·62	5·092	2·833
Min. steps/climb	0	0	0	0
Max. steps/climb	14	38	13	10
Steps/climb s.d.	2·257	7·062	2·244	1·317
Av. evals/climb	36·37	218·8	98·47	62·33
Min. evals/climb	17	17	17	17
Max. evals/climb	103	625	225	177
Evals/climb s.d.	12·50	112·9	35·91	21·07

points in the basins of all the peaks under SA is a set of 777,000 points. In an overall space of 25,600,000,000 points, this represents only just over 3×10^{-5} of the entire space.

Table 20 shows the result of reverse hillclimbing from the 48 known optimal TMs for the 4-state problem. The small number of different basin sizes for the algorithms is the result of redundancies in the representation. Once again, the four algorithms exhibit an ordering, by probability of reaching any peak on a single hillclimb, that should by now be becoming familiar. SA is a clear winner, AA is second, followed by MA and then LA. This pattern was also seen in the best five percent of peaks for the three NK landscapes (see Tables 8 to 16). In all three of those cases, SA ranked first and AA second. In two of them, MA was third and LA last and in one, LA third and MA last. This pattern will be seen in the three busy beaver landscapes. In fact, the pattern in the busy beaver landscapes is *exactly* that seen in the NK landscapes. SA is first in all three, AA is second in all three and MA beats LA on all but the simplest of the problems.

To examine whether the reverse hillclimbing results are accurate and to determine which algorithm can be expected to find peaks using the fewest evaluations, it is

Table 14. Expected and observed numbers of peaks and evaluations for 35 good NK 16,8 peaks.

	AA	LA	MA	SA
E(Peaks found)	1589	1019	1560	1862
Peaks found	1451	1042	1458	1854
E(Evals/peak)	229	2146	631	335
Evals/peak	251	2100	675	336

necessary to estimate the expected number of evaluations in a hillclimb, as was done in the NK landscape experiments. Table 21 shows the statistics obtained from 10,000 hillclimbs for each of the algorithms. Again we see a similar pattern to that seen for the NK problems. AA has the fewest expected evaluations and is followed by SA, MA and LA in that order. This relationship has a simple explanation, that was presented in §4.9.1(105). The most striking aspect of this table is the very small number of steps that a hillclimber can expect to take on this landscape before reaching a peak. For SA, the expectation is less than 2 steps.

Table 22 compares the actual number of peaks found and evaluations taken with the expected numbers. Because it is very difficult to locate these peaks with hillclimbing, each algorithm was run 20 million times to obtain a reasonable number of successes. The expected and observed numbers of peaks are in good agreement, as are the numbers of evaluations. In this case, the advantage that SA has over AA in terms of its probability of finding a peak outweighs its disadvantage in terms of expected number of evaluations per climb. This is the first time we have have seen the tradeoff resolved in favor of SA, and the victory is a very slight one. If we assume that the estimated number of evaluations per climb is actually correct, then the difference in the expected number of evaluations per peak location between the two algorithms is insignificant.

This experiment was repeated for peaks on the 3- and 2-state busy beaver problems. These results are shown in Tables 23 to 30. In both of these problems, SA again has the highest probability of peak location given a single climb, but is beaten by AA due to the high cost it incurs per uphill step.

Table 15. Reverse hillclimbing from the 180 peaks in a 16,4 NK landscape.

	AA	LA	MA	SA
# of basin sizes	179	140	178	158
Min. basin size	14,703	1	72	6
Max. basin size	55,146	5029	14,207	2565
Mean basin size	43,640	364	2761	364
Basin size s.d.	7944	588	2738	446
P(Discovery)	1·0	1·0	1·0	1·0
E(Climbs)	1	1	1	1

Table 16. Reverse hillclimbing from the best 9 of the 180 peaks in a 16,4 NK landscape.

	AA	LA	MA	SA
# of basin sizes	9	9	9	9
Min. basin size	42,817	69	4566	644
Max. basin size	54,927	5029	14,207	2485
Mean basin size	51,768	1161	7637	1444
Basin size s.d.	3723	1470	3499	634
P(Discovery)	0·165	0·160	0·151	0·198
E(Climbs)	6·076	6·269	6·618	5·042

Table 17. The result of 10,000 hillclimbs on the 16,4 NK landscape. The table shows data for the number of uphill steps taken for each algorithm and the number of evaluations performed per uphill walk.

	AA	LA	MA	SA
Av. steps/climb	6·064	15·20	6·622	3·886
Min. steps/climb	0	0	0	0
Max. steps/climb	19	44	19	10
Steps/climb s.d.	2·587	3·373	2·695	1·484
Av. evals/climb	41·58	260·2	122·9	79·18
Min. evals/climb	17	17	17	17
Max. evals/climb	126	721	321	177
Evals/climb s.d.	13·57	117·9	43·12	23·74

Table 18. Expected and observed numbers of peaks and evaluations for 9 good NK 16,4 peaks.

	AA	LA	MA	SA
E(Peaks found)	1645	1595	1511	1983
Peaks found	1530	1592	1545	2017
E(Evals/peak)	252	1631	813	399
Evals/peak	272	1634	795	393

Table 19. Reverse hillclimbing from 1000 random 4-state busy beaver peaks.

	AA	LA	MA	SA
# of basin sizes	470	440	320	356
Min. basin size	4	4	1	2
Max. basin size	545,848	360,864	42,970	61,606
Mean basin size	4305	2585	481	777
Basin size s.d.	27,647	16,521	2628	4282
P(Discovery)	0·0000075	0·0000066	0·0000071	0·0000085
E(Climbs)	133,754	151,713	141,453	117,993

4.10. Conclusion

This chapter introduced the reverse hillclimbing algorithm. This chapter showed that the algorithm makes it possible to determine the relative performance of two hillclimbing algorithms (that only take uphill steps) on a given problem. The algorithm also provides answers to several other questions concerning sizes of basins of attraction and lengths of paths to discover peaks. To answer these questions without reverse hillclimbing, it appears necessary to use statistical methods based on the occasional discovery of peaks. These methods are both computationally infeasible and of limited accuracy. Reverse hillclimbing, when it is useful, provides answers to the above questions very rapidly and with a high degree of precision. In many cases answers obtained with the method are exact.

The reverse hillclimbing algorithm cannot be used to discover high peaks on a fitness landscape. Rather, it is used as the basis for analysis once these peaks have been found by other methods. For instance, if a certain hillclimbing method has identified 100 peaks on a "rugged" landscape, reverse hillclimbing can be used to calculate the probability that other hillclimbers would find these peaks. If some other search algorithm has identified a good peak, reverse hillclimbing can produce the exact probability that a given form of hillclimbing would have found the point. This allows rapid and accurate comparisons without the need to run the hillclimber

Table 20. Reverse hillclimbing from the 48 peaks in the 4-state busy beaver problem.

	AA	LA	MA	SA
# of basin sizes	6	5	24	4
Min. basin size	10,420	3349	897	1426
Max. basin size	25,714	5648	2660	5364
Mean basin size	18,072	4511	1552	3370
Basin size s.d.	7637	1133	669	1936
P(Discovery)	0·000001169	0·000000610	0·000001002	0·000002012
E(Climbs)	855,538	1,639,043	997,524	497,129

to form an estimate of the probability. In previous work [110], this approach was used to assess the performance of a GA.

Reverse hillclimbing has the pleasing property of becoming more feasible as the landscape on which it is run gets more "rugged." That is, as basin sizes decrease (perhaps due to increased numbers of peaks, as we saw above with NK landscapes) reverse hillclimbing becomes increasingly competitive with sampling methods based on hillclimbing. On extremely difficult problems, reverse hillclimbing will perform extremely well since it never strays more than one vertex from the points in the basin of attraction during its descent. Therefore, the smaller the basin of attraction of a peak, the faster reverse hillclimbing will be and the more difficult it becomes to answer questions about the basin using other methods.

Other statistics about landscapes can also be easily investigated via reverse hillclimbing. These include the relationships between peak heights, basin sizes and probability of discovery. A trivial modification to the algorithm can be used to explore the "basin" of points *above* a certain point in a landscape. This allows calculation of statistics relating the number of uphill directions to the number of peaks that can be reached from a given point. This information might be used to direct the behavior of a hillclimber that expended variable effort in looking for uphill neighbors at each stage of the climb.

Table 21. The result of 10,000 hillclimbs on the 4-state busy beaver landscape. The table shows data for the number of uphill steps taken for each algorithm and the number of evaluations performed per uphill walk.

	AA	LA	MA	SA
Av. steps/climb	2·091	2·242	2·111	1·883
Min. steps/climb	0	0	0	0
Max. steps/climb	8	9	8	8
Steps/climb s.d.	1·316	1·461	1·307	1·165
Av. evals/climb	83·35	156·6	150·3	139·4
Min. evals/climb	49	49	49	49
Max. evals/climb	200	481	433	433
Evals/climb s.d.	25·82	70·14	62·73	55·92

Finally, reverse hillclimbing has highlighted the tradeoff between moving uphill quickly and moving in the steepest direction possible. The any ascent algorithm takes the first uphill direction it can find and consequently expends little energy looking for uphill directions. Steepest ascent always examines all neighbors and then chooses the steepest of these. Although, in our experience, steepest ascent always has a higher probability of finding a peak on a single climb (this can be shown exactly by reverse hillclimbing for a given problem), this advantage is usually not enough to compensate for the extra work it must do at each step. These two approaches represent extremes of behavior. Appendix B presents preliminary results of an attempt to find more balanced hillclimbers.

Table 22. Expected and observed numbers of peaks and evaluations for the 48 peaks on the 4-state busy beaver problem. The observed values are taken from 20 million hillclimbs with each of the four algorithms.

	AA	LA	MA	SA
E(Peaks found)	23	12	20	40
Peaks found	23	15	23	44
E(Evals/peak)	71,307,723	256,657,743	149,960,575	69,294,214
Evals/peak	71,270,514	208,393,004	130,625,970	63,161,171

Table 23. Reverse hillclimbing from 1000 random 3-state busy beaver peaks.

	AA	LA	MA	SA
# of basin sizes	287	279	212	224
Min. basin size	4	4	2	2
Max. basin size	20,219	13,616	7052	8142
Mean basin size	327	222	108	137
Basin size s.d.	1309	869	409	502
P(Discovery)	0·00368	0·00329	0·00364	0·00407
E(Climbs)	272	304	275	246

Table 24. Reverse hillclimbing from the 40 peaks in the 3-state busy beaver problem.

	AA	LA	MA	SA
# of basin sizes	13	14	22	14
Min. basin size	5659	2522	1247	2092
Max. basin size	20,219	13,616	7052	8142
Mean basin size	10,974	6702	3249	4156
Basin size s.d.	6013	4746	2183	2426
P(Discovery)	0·00399	0·00337	0·00381	0·00453
E(Climbs)	251	297	262	221

Table 25. The result of 10,000 hillclimbs on the 3-state busy beaver landscape. The table shows data for the number of uphill steps taken for each algorithm and the number of evaluations performed per uphill walk.

	AA	LA	MA	SA
Av. steps/climb	1·754	1·829	1·762	1·591
Min. steps/climb	0	0	0	0
Max. steps/climb	7	7	7	6
Steps/climb s.d.	1·114	1·189	1·116	1·005
Av. evals/climb	50·65	85·87	83·86	78·72
Min. evals/climb	31	31	31	31
Max. evals/climb	136	241	241	211
Evals/climb s.d.	15·64	35·68	33·47	30·14

Table 26. Expected and observed numbers of peaks and evaluations for the 48 peaks on the 3-state busy beaver problem. The observed values are taken from 10,000 hillclimbs with each of the four algorithms.

	AA	LA	MA	SA
E(Peaks found)	40	34	38	45
Peaks found	49	29	37	48
E(Evals/peak)	12,687	25,463	21,990	17,367
Evals/peak	12,662	29,609	22,665	16,400

Table 27. Reverse hillclimbing from 1000 random 2-state busy beaver peaks.

	AA	LA	MA	SA
# of basin sizes	37	35	27	27
Min. basin size	4	4	4	3
Max. basin size	372	292	200	274
Mean basin size	14·4	12·1	11·0	11·3
Basin size s.d.	26·4	20·7	14·2	19·0
P(Discovery)	0·409	0·393	0·406	0·425
E(Climbs)	2·448	2·545	2·460	2·355

Table 28. Reverse hillclimbing from the 4 peaks in the 2-state busy beaver problem.

	AA	LA	MA	SA
# of basin sizes	2	2	2	2
Min. basin size	370	290	198	273
Max. basin size	372	292	200	274
Mean basin size	371	291	199	273·5
Basin size s.d.	1	1	1	0·5
P(Discovery)	0·0354	0·0321	0·0304	0·0390
E(Climbs)	28·27	31·17	32·85	25·64

Table 29. The result of 10,000 hillclimbs on the 2-state busy beaver landscape. The table shows data for the number of uphill steps taken for each algorithm and the number of evaluations performed per uphill walk.

	AA	LA	MA	SA
Av. steps/climb	1·372	1·446	1·374	1·305
Min. steps/climb	0	0	0	0
Max. steps/climb	5	5	5	5
Steps/climb s.d.	0·917	0·980	0·911	0·838
Av. evals/climb	26·26	40·14	38·99	37·87
Min. evals/climb	17	17	17	17
Max. evals/climb	76	97	97	97
Evals/climb s.d.	7·898	15·68	14·57	13·41

Table 30. Expected and observed numbers of peaks and evaluations for the 48 peaks on the 2-state busy beaver problem.

	AA	LA	MA	SA
E(Peaks found)	354	321	304	390
Peaks found	356	317	290	364
E(Evals/peak)	742	1251	1281	971
Evals/peak	737	1266	1344	1040