Explaining Optimization In Genetic Algorithms with Uniform Crossover

Keki M Burjorjee
Zite, Inc.
487 Bryant St.
San Francisco, CA 94107
kekib@cs.brandeis.edu

ABSTRACT

Hyperclimbing is an intuitive, general-purpose, global optimization heuristic applicable to discrete product spaces with rugged or stochastic cost functions. The strength of this heuristic lies in its insusceptibility to local optima when the cost function is deterministic, and its tolerance for noise when the cost function is stochastic. Hyperclimbing works by *decimating* a search space, i.e., by iteratively fixing the values of small numbers of variables. The *hyperclimbing hypothesis* posits that genetic algorithms with uniform crossover (UGAs) perform optimization by implementing *efficient* hyperclimbing. Proof of concept for the hyperclimbing hypothesis comes from the use of an analytic technique that exploits algorithmic symmetry. By way of validation, we present experimental results showing that a simple tweak inspired by the hyperclimbing hypothesis dramatically improves the performance of a UGA on large, random instances of MAX-3SAT and the Sherrington Kirkpatrick Spin Glasses problem. An exciting corollary of the hyperclimbing hypothesis is that a form of *implicit parallelism* more powerful than the kind described by Holland underlies optimization in UGAs. The implications of the hyperclimbing hypothesis for Evolutionary Computation and Artificial Intelligence are discussed.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence—*Problem Solving, Control Methods, and Search*; F.2 [**Theory of Computation**]: Analysis of Algorithms And Problem Complexity—*Miscellaneous*

General Terms

Algorithms; Theory

Keywords

Genetic Algorithms; Uniform Crossover; Hyperclimbing ; MAXSAT; Spin Glasses; Global Optimization; Decimation

FOGA'13, January 16Ð20, 2013, Adelaide, Australia.

1. INTRODUCTION

Optimization in genetic algorithms with uniform crossover (UGAs) is one of the deep mysteries of Evolutionary Computation. The use of uniform crossover causes genetic loci to be unlinked, i.e. recombine freely. This form of recombination was first used by Ackley [1] in 1987, and was subsequently studied by Syswerda [29], Eshelman et al. [8], and Spears & De Jong [28, 7], who found that it frequently outperformed crossover operators that induce tight linkage between genetic loci (e.g. one point crossover). It is generally acknowledged that the efficacy of uniform crossover, a highly disruptive form of variation, cannot be explained within the rubric of the *building block hypothesis* [11, 25, 9], the beleaguered, but still influential explanation for optimization in genetic algorithms with strong linkage between loci. Yet, no alternate, scientifically rigorous explanation for optimization in UGAs has been proposed. The hypothesis presented in this paper addresses this gap. This hypothesis posits that UGAs perform optimization by implicitly and efficiently implementing a global search heuristic called *hyperclimbing*.

Hyperclimbing is a *global decimation heuristic*, and as such is in good company. Global decimation heuristics are currently the state of the art approach to solving large instances of the Boolean Satisfiablity Problem (SAT) close to the SAT/UNSAT threshhold (i.e. hard instances of SAT) [18]. Conventional global decimation heuristics—e.g. Survey Propagation [20], Belief Propagation, Warning Propagation [3]—use message passing algorithms to compile statistical information about the space being searched. This information is then used to irrevocably fix the values of one, or a small number, of search space attributes, effectively reducing the size of the space. The decimation heuristic is then recursively applied to the resulting search space. Survey Propagation, perhaps the best known global decimation strategy, has been used along with *Walksat* [27] to solve instances of SAT with upwards of a million variables. The hyperclimbing hypothesis posits that in practice, UGAs also perform optimization by decimating the search spaces to which they are applied. Unlike conventional decimation strategies, however, a UGA obtains statistical information about the search space implicitly and efficiently by means other than message passing.

We stress at the outset that our main concern in this paper is scientific rigor in the Popperian tradition [24], not mathematical proof within a formal axiomatic system. To be considered scientifically rigorous, a hypothesis about an evolutionary algorithm should meet at least the following two criteria: First, it should be based on weak assumptions

about the distribution of fitness induced by the ad-hoc representational choices of evolutionary algorithm users. This is nothing but an application of Occam's Razor to the domain of Evolutionary Computation. Second, the hypothesis should predict unexpected behavior. (Popper noted that the predictions that lend the most credence to a scientific hypothesis are the ones that augur phenomena that would not be expected in the absence of the hypothesis—e.g. gravitational lensing in the case of Einstein's theory of General Relativity)

The criteria above constitute the most basic requirements that a hypothesis should meet. But one can ask for more; after all, one has greater control over evolutionary algorithms than one does over, say, gravity. Recognizing this advantage, we specify two additional criteria. The first is upfront proof of concept. Any predicted behavior must be demonstrated unambiguously, even if it is only on a contrived fitness function. Requiring upfront proof of concept heads off a situation in which predicted behavior fails to materialize in the setting where it is most expected (cf. Royal Roads experiments [21]). Such episodes tarnish not just the hypothesis concerned but the scientific approach in general—an approach, it needs to be said in light of the current slant of theoretical research in evolutionary computation—that lies at the foundation of many a vibrant field of engineering. The second criterion is upfront validation of unexpected behavior on a non-contrived fitness function. Given the control we have over an evolutionary algorithm, it is reasonable to ask for a prediction of unexpected behavior on a real-world fitness function, and to require upfront validation of this prediction.

The hyperclimbing hypothesis, we are pleased to report, meets all of the criteria listed above. The rest of this paper is organized as follows: Section 2 provides an informal description of the hyperclimbing heuristic and lists the underlying assumptions about the distribution of fitness. A more formal description of the hyperclimbing heuristic appears in Appendix A. Section 3 outlines symmetries of uniform crossover and length independent mutation that we subsequently exploit. Section 4, presents proof of concept, i.e. it describes a stochastic fitness function—the Royal Roads of the hyperclimbing hypothesis—on which a UGA behaves as described. Then, by exploiting the symmetries of uniform crossover and length independent mutation, we argue that the adaptive capacity of a UGA scales extraordinarily well as the size of the search space increases. We follow up with experimental tests that validate this conclusion. In section 5 we make a prediction about the behavior of a UGA, and validate this prediction on large, randomly generated instances of MAX-3SAT and the Sherrington Kirkpatric Spin Glasses problem. We conclude in Section 6 with a discussion about the generalizability of the hyperclimbing hypothesis and its implications for Evolutionary Computation.

2. THE HYPERCLIMBING HEURISTIC

For a sketch of the hyperclimbing heuristic, consider a search space $S = \{0,1\}^\ell$, and a (possibly stochastic) fitness function that maps points in S to real values. Given some index set $\mathcal{I} \subseteq \{1,\ldots,\ell\}$, $\mathcal{I}$ partitions S into $2^{|\mathcal{I}|}$ subsets called *schemata* (singular schema) [21] as in the following example: suppose $\ell = 4$, and $\mathcal{I} = \{1,3\}$, then $\mathcal{I}$ partitions S into the subsets $\{0000, 0001, 0100, 0101\}$, $\{0010, 0011, 0110, 0111\}$, $\{1000, 1001, 1100, 1101\}$, $\{1010, 1011, 1110, 1111\}$. Partitions of this type are called *schema partitions*. Schemata and schema partitions can also be expressed using templates, for example, $0*1*$ and $\#*\#*$ respectively. Here the symbol $*$ stands for 'wildcard', and the symbol $\#$ denotes a defined bit. The *order* of a schema partition is simply the cardinality of the index set that defines the partition. Clearly, schema partitions of lower order are coarser than schema partitions of higher order. The *effect* of a schema partition is defined to be the variance of the expected fitness of the constituent schemata under sampling from the uniform distribution over each schema. So for example, the effect of the schema partition $\#*\#* = \{0*0*, 0*1*, 1*0*, 1*1*\}$ is

$$\frac{1}{4}\sum_{i=0}^{1}\sum_{j=0}^{1}(F(i*j*) - F(****))^2$$

where the operator F gives the expected fitness of a schema under sampling from the uniform distribution.

A hyperclimbing heuristic starts by sampling from the uniform distribution over the entire search space. It subsequently identifies a coarse schema partition with a non-zero effect, and limits future sampling to a schema in this partition with above average expected fitness. In other words the hyperclimbing heuristic fixes the *defining bits* [21] of this schema in the population. This schema constitutes a new (smaller) search space to which the hyperclimbing heuristic is recursively applied. Crucially, the act of fixing defining bits in a population has the potential to "generate" a detectable non-zero effects in schema partitions that previously might have had a negligible effects. For example, the schema partition $*\#****\#$ may have a negligible effect, whereas the schema partition $1\#*0*\#$ has a detectable non-zero effect. This observation is essential to understanding the hyperclimbing heuristic's capacity for optimization. A fitness distribution in which this structure is recursively repeated is said to have *staggered conditional effects*. The assumption that a fitness function induces staggered conditional effects is a weak assumption. In comparison, the building block hypothesis assumes unstaggered unconditional effects, and even this only when the defining bits of building blocks can be unlinked. This is a much stronger assumption because there are vastly more ways for effects to be staggered and conditional than unstaggered and unconditional. A more formal description of the hyperclimbing heuristic can be found in Appendix A, and a simple realization of a fitness function with staggered conditional effects appears in Section 4

At each step in its progression, hyperclimbing is sensitive, not to the fitness value of any individual point, but to the sampling means of relatively coarse schemata. This heuristic is, therefore, *natively* able to tackle optimization problems with stochastic cost functions. Considering its simplicity, the hyperclimbing heuristic has almost certainly been lighted upon by other researchers in the general field of discrete optimization. In all likelihood it was set aside each time because of the seemingly high cost of implementation for all but the smallest of search spaces or the coarsest of schema partitions. Given a search space comprised of ℓ binary variables, there are $\binom{\ell}{o}$ schema partitions of order o. For any fixed value of o, $\binom{\ell}{o} \in \Omega(\ell^o)$ [6]. The exciting finding presented in this paper is that UGAs can implement hyperclimbing cheaply for large values of ℓ, and values of o that are small, but greater than one.

3. SYMMETRIES OF A UGA

A genetic algorithm with a finite but non-unitary population of size N (the kind of GA used in practice) can be modeled by a Markov Chain over a state space consisting of all possible populations of size N [22]. Such models tend to be unwieldy [13] and difficult to analyze for all but the most trivial fitness functions. Fortunately, it is possible to avoid this kind of modeling and analysis, and still obtain precise results for non-trivial fitness functions by exploiting some simple symmetries introduced through the use of uniform crossover and length independent mutation.

A homologous crossover operator between two chromosomes of length ℓ can be modeled by a vector of ℓ random binary variables $\langle X_1, \ldots, X_\ell \rangle$ from which crossover masks are sampled. Likewise, a mutation operator can be modeled by a vector of ℓ random binary variables $\langle Y_1, \ldots, Y_\ell \rangle$ from which mutation masks are sampled. Only in the case of uniform crossover are the random variables $X_1, \ldots, X_\ell$ independent and identically distributed. This absence of *positional bias* [8] in uniform crossover constitutes a symmetry. Essentially, permuting the bits of all chromosomes using some permutation π before crossover, and permuting the bits back using π^{-1} after crossover has no effect on the dynamics of a UGA. If, in addition, the random variables $Y_1, \ldots, Y_\ell$ that model the mutation operator are independent and identically distributed (which is typical), and (more crucially) independent of the value of ℓ, then in the event that the values of chromosomes at some locus i are immaterial during fitness evaluation, the locus i can be "spliced out" without affecting allele dynamics at other loci. In other words, the dynamics of the UGA can be *coarse-grained* [4].

These conclusions flow readily from an appreciation of the symmetries induced by uniform crossover and length independent mutation. While the use of symmetry arguments is uncommon in EC research, symmetry arguments form a crucial part of the foundations of physics and chemistry. Indeed, according to the theoretical physicist E. T. Jaynes "almost the only known exact results in atomic and nuclear structure are those which we can deduce by symmetry arguments, using the methods of group theory" [16, p331-332]. Note that the conclusions above hold true regardless of the selection scheme (fitness proportionate, tournament, truncation, etc), and any fitness scaling that may occur (sigma scaling, linear scaling etc). "The great power of symmetry arguments lies just in the fact that they are not deterred by any amount of complication in the details", writes Jaynes [16, p331]. An appeal to symmetry, in other words, allows one to cut through complications that might hobble attempts to reason within a formal axiomatic system.

Of course, symmetry arguments are not without peril. However, when used sparingly and only in circumstances where the symmetries are readily apparent, they can yield significant insight at low cost. It bears emphasizing that the goal of foundational work in evolutionary computation is not pristine mathematics within a formal axiomatic system, but insights of the kind that allow one to a) explain optimization in current evolutionary algorithms on real world problems, and b) design more effective evolutionary algorithms.

4. PROOF OF CONCEPT

Providing unambiguous evidence that a UGA can behave as described in the hyperclimbing hypothesis is one of the explicit goals of this paper. To achieve this aim we introduce the *staircase function*, a "Royal Roads" for the hyperclimbing heuristic, and provide experimental evidence that a UGA can perform hyperclimbing on a particular parameterization of this function. Then, using symmetry arguments, we conclude that the running time and the number of fitness queries required to achieve equivalent results scale surprisingly well with changes to key parameters. An experimental test validates this conclusion.

Algorithm 1:
A staircase function with descriptor $(h, o, \delta, \sigma, \ell, L, V)$

Input: g is a chromosome of length ℓ

```
x ← some value drawn from the distribution N(0, 1)
for i ← 1 to h do
    if Ξ_{L_{i:}}(g) = V_{i1} ... V_{io} then
        x ← x + δ
    else
        x ← x − (δ/(2^o − 1))
        break
    end
end
return x
```

DEFINITION 1. *A staircase function descriptor is a 6-tuple $(h, o, \delta, \ell, L, V)$ where h, o and ℓ are positive integers such that $ho \leq \ell$, δ is a positive real number, and L and V are matrices with h rows and o columns such that the values of V are binary digits, and the elements of L are distinct integers in $\{1, \ldots, \ell\}$.*

For any positive integer ℓ, let $[\ell]$ denote the set $\{1, \ldots, \ell\}$, and let $\mathfrak{B}_\ell$ denote the set of binary strings of length ℓ. Given any k-tuple, x, of integers in $[\ell]$, and any binary string $g \in \mathfrak{B}_\ell$, let $\Xi_x(g)$ denote the string $b_1, \ldots, b_k$ such that for any $i \in [k]$, $b_i = g_{x_i}$. For any $m \times n$ matrix M, and any $i \in [m]$, let $M_{i:}$ denote the n-tuple that is the i^{th} row of M. Let $\mathcal{N}(a, b)$ denote the normal distribution with mean a and variance b. Then the function, f, described by the staircase function descriptor $(h, o, \delta, \ell, L, V)$ is the stochastic function over the set of binary strings of length ℓ given by Algorithm 1. The parameters h, o, δ, and ℓ are called the *height, order, increment* and *span*, respectively, of f. For any $i \in [h]$, we define *step* i of f to be the schema $\{g \in \mathfrak{B}_\ell | \Xi_{L_{i:}}(g) = V_{i1} \ldots V_{io}\}$, and define *stage* i of f to be the schema $\{g \in \mathfrak{B}_\ell | (\Xi_{L_{1:}}(g) = V_{11} \ldots V_{1o}) \wedge \ldots \wedge (\Xi_{L_{i:}}(g) = V_{i1} \ldots V_{io})\}$.

The stages of a staircase function can be visualized as a progression of nested *hyperplanes*[1], with hyperplanes of higher order and higher expected fitness nested within hyperplanes of lower order and lower expected fitness. By choosing an appropriate scheme for mapping a high-dimensional hypercube onto a two dimensional plot, it becomes possible to visualize this progression of hyperplanes in two dimensions (Appendix B).

A step of the staircase function is said to have been *climbed* when future sampling of the search space is largely limited to that step. Just as it is hard to climb higher steps of a physical staircase without climbing lower steps first, it

[1]A hyperplane, in the current context, is just a geometrical representation of a schema [10, p 53].

Algorithm 2: Pseudocode for the UGA used. The population size is an even number, denoted N, the length of the chromosomes is ℓ, and for any chromosomal bit, the probability that the bit will be flipped during mutation (the per bit mutation probability) is p_m. The population is represented internally as an N by ℓ array of bits, with each row representing a single chromosome. GENERATE-UX-MASKS(x, y) creates an x by y array of bits drawn from the uniform distribution over $\{0, 1\}$. GENERATE-MUT-MASKS(x, y, z) returns an x by y array of bits such that any given bit is 1 with probability z.

```
pop ← INITIALIZE-POPULATION(N,ℓ)
while some termination condition is unreached do
    fitnessValues ← EVALUATE-FITNESS(pop)
    adjustedFitVals ← SIGMA-SCALE(fitnessValues)
    parents ← SUS-SELECTION(pop, adjustedFitVals)
    crossMasks ←GENERATE-UX-MASKS(N/2, ℓ)
    for i ← 1 to N/2 do
        for j ← 1 to ℓ do
            if crossMasks[i, j] = 0 then
                newPop[i, j] ← parents[i, j]
                newPop[i + N/2, j] ← parents[i + N/2, j]
            else
                newPop[i, j] ← parents[i + N/2, j]
                newPop[i + N/2, j] ← parents[i, j]
            end
        end
    end
    mutMasks ←GENERATE-MUT-MASKS(N, ℓ, p_m)
    for i ← 1 to N do
        for j ← 1 to ℓ do
            newPop[i, j] ← XOR(newPop[i, j],
            mutMasks[i, j])
        end
    end
    pop ← newPop
end
```

can be computationally expensive to identify higher steps of a staircase function without identifying lower steps first (Theorem 1, Appendix C). The difficulty of climbing step $i \in [h]$ *given* stage $i-1$, however, is non-increasing with respect to i (Corollary 1, Appendix C). We conjecture that staircase functions capture a feature— *staggered conditional effects*— that is widespread within the fitness functions resulting from the representational choices of GA users.

4.1 UGA Specification

The pseudocode for the UGA used in this paper is given in Algorithm 2. The free parameters of the UGA are N (the size of the population), p_m (the per bit mutation probability), and EVALUATE-FITNESS (the fitness function). Once these parameters are fixed, the UGA is fully specified. The specification of a fitness function implicitly determines the length of the chromosomes, ℓ. Two points deserve further elaboration:

1. The function SUS-SELECTION takes a population of size N, and a corresponding set of fitness values as inputs. It returns a set of N parents drawn by fitness proportionate *stochastic universal sampling* (SUS). Instead of selecting N parents by spinning a roulette wheel with one pointer N times, stochastic universal sampling selects N parents by spinning a roulette wheel with N equally spaced pointers just once. Selecting parents this way has been shown to reduce sampling error [2, 21].

2. When selection is fitness proportionate, an increase in the average fitness of the population causes a decrease in selection pressure. The UGA in Algorithm 2 combats this ill-effect by using sigma scaling [21, p 167] to adjust the fitness values returned by EVALUATE-FITNESS. These adjusted fitness values, not the raw ones, are used when selecting parents. Let $f_x^{(t)}$ denote the raw fitness of some chromosome x in some generation t, and let $\overline{f^{(t)}}$ and $\sigma^{(t)}$ denote the mean and standard deviation of the raw fitness values in generation t respectively. Then the *adjusted fitness* of x in generation t is given by $h_x^{(t)}$ where, if $\sigma^{(t)} = 0$ then $h_x^{(t)} = 1$, otherwise,

$$h_x^{(t)} = \min(0, 1 + \frac{f_x^{(t)} - \overline{f^{(t)}}}{\sigma^{(t)}})$$

The use of sigma scaling also causes negative fitness values to be handled appropriately.

.

4.2 Performance of a UGA on a class of Staircase Functions

Let f be a staircase function with descriptor $(h, o, \delta, \ell, L, V)$, we say that f is *basic* if $\ell = ho$, $L_{ij} = o(i-1) + j$, (i.e. if L is the matrix of integers from 1 to ho laid out row-wise), and V is a matrix of ones. If f is known to be basic, then the last three elements of the descriptor of f are fully determinable from the first three, and its descriptor can be shortened to (h, o, δ). Given some staircase function f with descriptor $(h, o, \delta, \ell, L, V)$, we define the *basic form* of f to be the (basic) staircase function with descriptor (h, o, δ).

Let ϕ^* be the basic staircase function with descriptor $(h = 50, o = 4, \delta = 0.3)$, and let U denote the UGA defined in section 4.1 with a population size of 500, and a per bit mutation probability of 0.003 (i.e, $p_m = 0.003$). Figure 1a shows that U is capable of robust optimization when applied to ϕ^* (We denote the resulting algorithm by U^{ϕ^*}). Figure 1c shows that under the action of U, the first four steps of ϕ^* go to fixation[2] in ascending order. When a step gets fixed, future sampling will largely be confined to that step—in effect, the hyperplane associated with the step has been climbed. Note that the UGA does not need to "fully" climb a step before it begins climbing the subsequent step (Figure 1c). Animation 1 in the online appendix[3] shows that the hyperclimbing behavior of U^{ϕ^*} continues beyond the first four steps.

[2]The terms 'fixation' and 'fixing' are used loosely here. Clearly, as long as the mutation rate is non-zero, no locus can ever be said to go to fixation in the strict sense of the word.

[3]Online appendix available at http://bit.ly/QFHNAk

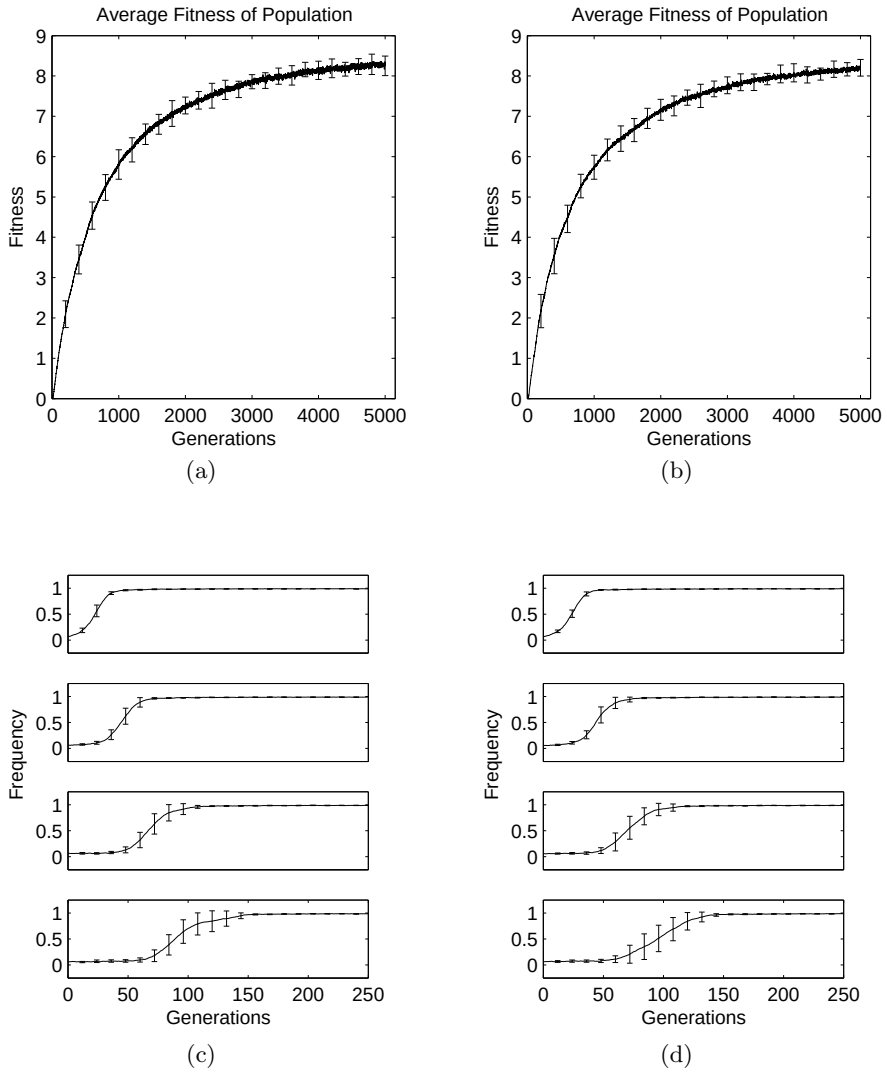


Figure 1: (a) The mean, across 20 trials, of the average fitness of the population of U^{ϕ^*} in each of 5000 generations. The error bars show five standard errors above and below the mean every 200 generations. (c) Going from the top plot to the bottom plot, the mean frequencies, across 20 trials, of the first four steps of the staircase function U^{ϕ^*} in each of the first 250 generations. The error bars show three standard errors above and below the mean every 12 generations. (b,d) Same as the plots on the left, but for U^{ϕ}

4.3 Symmetry Analysis and Experimental Confirmation

Let W be some UGA. For any staircase function f, and any $x \in [0,1]$, let $p^{(t)}_{(W^f,i)}(x)$ denote the probability that the frequency of *stage* i of f in generation t of W^f is x. Let f^* be the basic form of f. Then, by appreciating the symmetries between the UGAs W^f and W^{f^*} one can conclude the following:

CONCLUSION 1. *For any generation t, any $i \in [h]$, and any $x \in [0,1]$, $p^{(t)}_{(W^f,i)}(x) = p^{(t)}_{(W^{f^*},i)}(x)$*

This conclusion straightforwardly entails that to raise the average fitness of a population by some attainable value,

1. The expected number of generations required is *constant* with respect to the span of a staircase function (i.e., the query complexity is constant)
2. The running time[4] scales *linearly* with the span of a staircase function
3. The running time and the number of generations are unaffected by the last two elements of the descriptor of a staircase function

Let f be some staircase function with basic form ϕ^* (defined in Section 4.2). Then, given the above, the application of U to f should, discounting deviations due to sampling,

[4]Here, we mean the running time in the conventional sense, not the number of fitness queries.

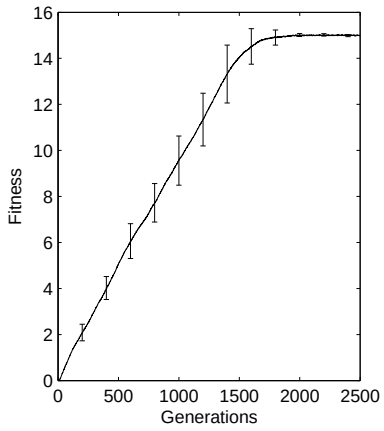
produce results identical to those shown in Figures 1a and 1c. We validated this "corollary" by applying U to the staircase function ϕ with descriptor $(h = 50, o = 4, \delta = 0.3, \ell = 20000, L, V)$ where L and V were randomly generated. The results are shown in Figures 1b and 1d. Note that gross changes to the matrices L and V, and an increase in the span of the staircase function by two orders of magnitude did not produce any statistically significant changes. It is hard to think of another algorithm with better scaling properties on this non-trivial class of fitness functions.

5. VALIDATION

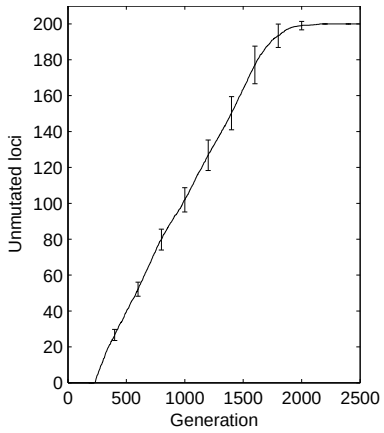
Let us pause to consider a curious aspect of the behavior of U^{ϕ^*}. Figure 1 shows that the growth rate of the average fitness of the population of U^{ϕ^*} decreases as evolution proceeds, and the average fitness of the population plateaus at a level that falls significantly short of the maximum expected average population fitness of 15. As discussed in the previous section, the difficulty of climbing step i given stage $i-1$ is non-increasing with respect to i. So, given that U successfully identifies the first step of ϕ^*, why does it fail to identify all remaining steps? To understand why, consider some binary string that belongs to the i^{th} stage of ϕ^*. Since the mutation rate of U is 0.003, the probability that this binary string will still belong to stage i *after* mutation is 0.997^{io}. This entails that as i increases, U^{ϕ^*} is less able to "hold" a population within stage i. In light of this observation, one can infer that as i increases the sensitivity of U to the conditional fitness signal of step i given stage $i-1$ will decrease. This loss in sensitivity explains the decrease in the growth rate of the average fitness of U^{ϕ^*}. We call the "wastage" of fitness queries described here *mutational drag*.

To curb mutational drag in UGAs, we conceived of a very simple tweak called *clamping*. This tweak relies on parameters `flagFreqThreshold` $\in [0.5, 1]$, `unflagFreqThreshold` $\in [0.5, $`flagFreqThreshold`$]$, and the positive integer `waitingPeriod`. If the *one-frequency* or the *zero-frequency* of some locus (i.e. the frequency of the bit 1 or the frequency of the bit 0, respectively, at that locus) at the beginning of some generation is greater than `flagFreqThreshold`, then the locus is flagged. Once flagged, a locus remains flagged as long as the one-frequency or the zero-frequency of the locus is greater than `unflagFreqThreshold` at the beginning of each subsequent generation. If a flagged locus in some generation t has remained constantly flagged for the last `waitingPeriod` generations, then the locus is considered to have passed our fixation test, and is not mutated in generation t. This tweak is called clamping because it is expected that in the absence of mutation, a locus that has passed our fixation test will quickly go to strict fixation, i.e. the one-frequency, or the zero-frequency of the locus will get "clamped" at one for the remainder of the run.

Let U_c denote a UGA that uses the clamping mechanism described above and is identical to the UGA U in every other way. The clamping mechanism used by U_c is parameterized as follows: `flagFreqThreshold` $= 0.99$, `unflagFreqThreshold` $= 0.9$, `waitingPeriod=200`. The performance of $U_c^{\phi^*}$ is displayed in figure 2a. Figure 2b shows the number of loci that the clamping mechanism left unmutated in each generation. These two figures show that the clamping mechanism effectively allowed U_c to climb all the stages of ϕ^*. Animation 2 in the online appendix shows the



(a)



(b)

Figure 2: *(Top)* The mean (across 20 trials) of the average fitness of the UGA U_c on the staircase function ϕ^*. Errorbars show five standard errors above and below the mean every 200 generations. *(Bottom)* The mean (across 20 trials) of the number of loci left unmutated by the clamping mechanism. Errorbars show three standard errors above and below the mean every 200 generations

one-frequency dynamics, across 500 generations, of a single run of $U_c^{\phi^*}$. The action of the clamping mechanism can be seen in the absence of 'jitter' in the one-frequencies of loci that have been at fixation for 200 or more generations.

If the hyperclimbing hypothesis is accurate, then mutational drag is likely to be an issue when UGAs are applied to other problems, especially large instances that require the use of long chromosomes. In such cases, the use of clamping should improve performance. We now present the results of experiments where the use of clamping clearly improves the performance of a UGA on large instances of MAX-3SAT and the Sherrington Kirkpatrik Spin Glasses problem.

5.1 Validation on MAX-3SAT

MAX-kSAT [14] is one of the most extensively studied combinatorial optimization problems. An instance of this problem consists of n boolean variables, and m clauses. The

literals of the instance are the n variables and their negations. Each clause is a disjunction of k of the total possible $2n$ literals. Given some MAX-kSAT instance, the value of a particular setting of the n variables is simply the number of the m clauses that evaluate to *true*. In a *uniform random* MAX-kSAT problem, the clauses are generated by picking each literal at random (with replacement) from amongst the $2n$ literals. Generated clauses containing multiple copies of a variable, and ones containing a variable and its negation, are discarded and replaced.

Let Q denote the UGA defined in section 4.1 with a population size of 200 ($N = 200$) and a per bit mutation probability of 0.01 (i.e., $p_m = 0.01$). We applied Q to a randomly generated instance of the Uniform Random 3SAT problem, denoted *sat*, with 1000 binary variables and 4000 clauses. Variable assignments were straightforwardly encoded, with each bit in a chromosome representing the value of a single variable. The fitness of a chromosome was simply the number of clauses satisfied under the variable assignment represented. Figure 3a shows the average fitness of the population of Q^{sat} over 7000 generations. Note that the growth in the maximum and average fitness of the population tapered off by generation 1000.

The UGA Q was applied to *sat* once again; this time, however, the clamping mechanism described above was activated in generation 2000. The resulting UGA is denoted Q_c^{sat}. The clamping parameters used were as follows: `flagFreqThreshold` = 0.99, `unflagFreqthreshold` = 0.8, `waitingPeriod` = 200. The average fitness of the population of Q_c^{sat} over 7000 generations is shown in Figure 3b, and the number of loci that the clamping mechanism left unmutated in each generation is shown in Figure 3c. Once again, the growth in the maximum and average fitness of the population tapered off by generation 1000. However, the maximum and average fitness began to grow once again starting at generation 2200. This growth coincides with the commencement of the clamping of loci (compare Figures 3b and 3c).

5.2 Validation on an SK Spin Glasses System

A Sherrington Kirkpatrick Spin Glasses system is a set of coupling constants J_{ij}, with $1 \leq i < j \leq \ell$. Given a configuration of "spins" $(\sigma_1, \ldots, \sigma_\ell)$, where each spin is a value in $\{+1, -1\}$, the "energy" of the system is given by

$$E(\sigma) = -\sum_{1 \leq i < j \leq \ell} J_{ij}\sigma_i\sigma_j$$

The goal is to find a spin configuration that minimizes energy. By defining the fitness of some spin configuration σ to be $-E(\sigma)$ we remain true to the conventional goal in genetic algorithmics of maximizing fitness. The coupling constants in J may be drawn from the set $\{-1, 0, +1\}$ or from the gaussian distribution $\mathcal{N}(0, 1)$. Following Pelikan et al. [23], we used coupling constants drawn from $\mathcal{N}(0, 1)$. Each chromosome in the evolving population straightforwardly represented a spin configuration, with the bits 1 and 0 denoting the spins $+1$ and -1 respectively[5]. The UGAs Q and Q_c (described in the previous subsection) were applied to a randomly generated Sherrington Kirkpatrik spin glasses system over 1000 spins, denoted *spin*. The results obtained (Figures 3d, 3e, and 3f) were similar to the results described in the previous subsection.

It should be noted that clamping by itself does not cause decimation. It merely enforces strict fixation once a high degree of fixation has already occurred along some dimension. In other words, clamping can be viewed as a decimation "lock-in" mechanism as opposed to a decimation "forcing" mechanism. Thus, the occurrence of clamping shown in Figure 3 entails the prior occurrence of decimation.[6]

The effectiveness of clamping demonstrated in this section lends considerable support to the hyperclimbing hypothesis. The method followed is out of Popper's Logic of Scientific Discovery [24]. A *scientific* theory allows one to make testable predictions of the form "if experiment X is executed, outcome Y will be observed". One is free to choose any X and Y as long as X entails Y given the theory. If the test is successful, the theory gains credibility in proportion to the extent to which Y is unanticipated in the absence of the theory. More support of this kind can be found in the work of Huifang and Mo [15] where the use of clamping improved the performance of a UGA on a completely different problem—optimizing the weights of a quantum neural network.

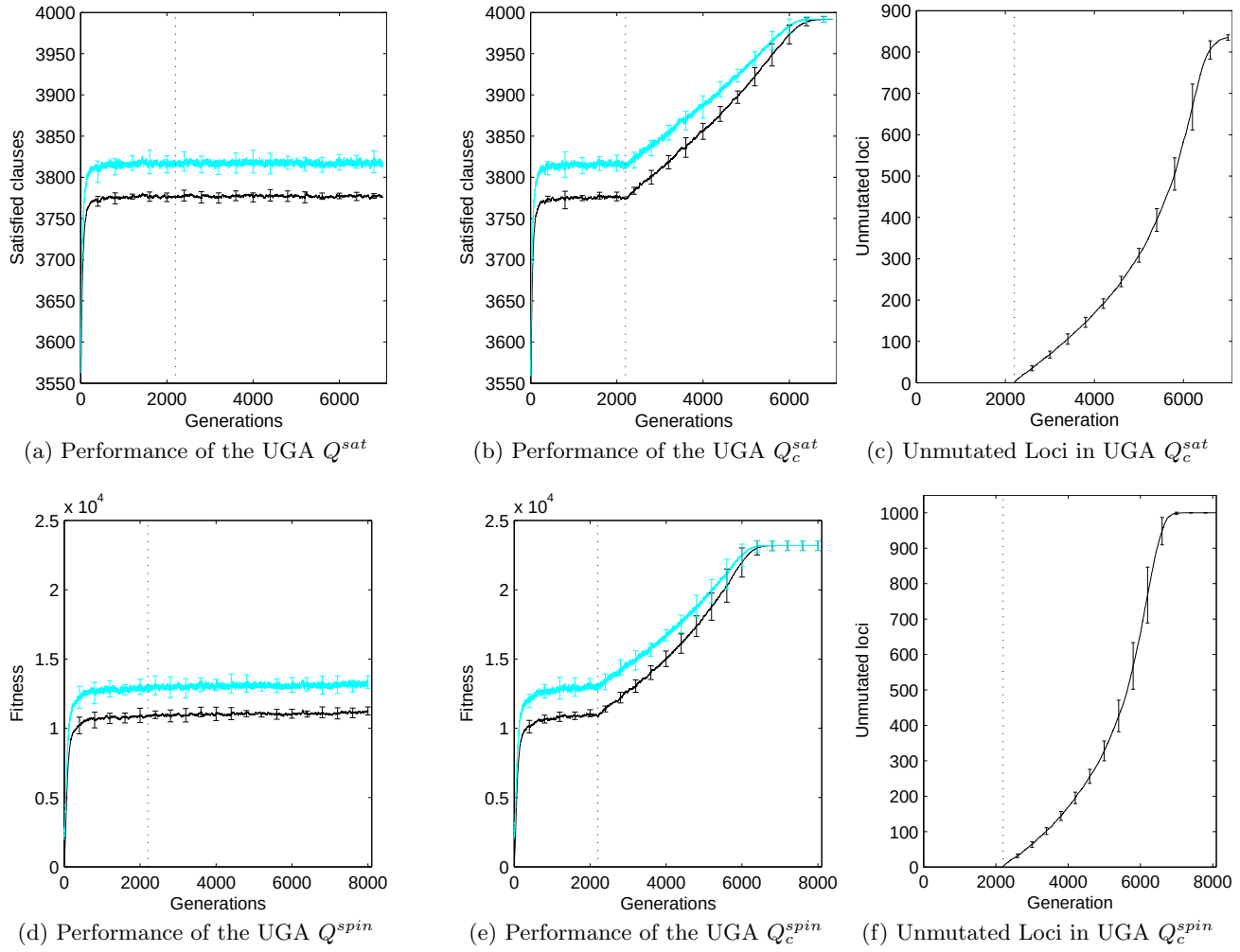
6. CONCLUSION

Simple genetic algorithms with uniform crossover (UGAs) perform optimization by implicitly exploiting the structure of fitness distributions that arise in practice through the ad-hoc representational choices of users. Two key questions are i) What is the nature of this structure? and ii) How is this structure exploited by the UGA? This paper offers a hypothesis that answers these and other questions about UGA behavior. The submitted hypothesis satisfies two basic requirements that one might expect any scientific hypothesis to meet—it relies only on assumptions that are weak, and it predicts an unexpected phenomenon. The hypothesis meets two additional requirements specific to the domain of evolutionary computation: It is accompanied by upfront proof of concept, and upfront validation. Section 4 unambiguously showed that a UGA can behave as described in the hyperclimbing hypothesis, and in Section 5, we predicted behavior that would not be expected in the absence of the hyperclimbing hypothesis, and validated this behavior on two non-contrived fitness functions: MAX-3SAT and the Sherrington Kirkpatrick Spin Glasses Problem.

An exciting corollary of the hyperclimbing hypothesis is that *implicit parallelism is real*. To be sure, what we mean

[5]Given an $n \times \ell$ matrix P representing a population of n spin configurations, each of size ℓ, the energies of the spin configurations can be expressed compactly as $-PJ^TP^T$ where J is an $\ell \times \ell$ upper triangular matrix containing the coupling constants of the SK system.

[6]A cautionary note: It may be tempting, based on the results obtained, to speculate that mutation hurts UGA performance, either on the fitness functions examined, or in general. After all, if one stops using mutation altogether, then the problem described at the beginning of Section 5—the problem addressed by clamping—disappears. We stress that this would be an incorrect conclusion to draw. A rigorous treatment of the specific roles played by mutation and uniform crossover in the implementation of hyperclimbing is beyond our current scope. We emphasize, however, that they *both* have parts to play. Briefly, mutation prevents the strict fixation of loci that have lost entropy to random drift, and uniform crossover allows hyperclimbing to proceed in parallel [5, Chapter 4].



(a) Performance of the UGA Q^{sat} (b) Performance of the UGA Q_c^{sat} (c) Unmutated Loci in UGA Q_c^{sat}

(d) Performance of the UGA Q^{spin} (e) Performance of the UGA Q_c^{spin} (f) Unmutated Loci in UGA Q_c^{spin}

Figure 3: ***(a,b)*** **The performance, over 10 trials, of the UGAs Q and the UGA Q_c on a randomly generated instance of the Uniform Random 3SAT problem with 1000 variables and 4000 clauses. The mean (across trials) of the average fitness of the population is shown in black. The mean of the best-of-population fitness is shown in blue. Errorbars show five standard errors above and below the mean every 400 generations.** ***(c)*** **The mean number of loci left unmutated by the clamping mechanism used by Q_c. Errorbars show three standard errors above and below the mean every 400 generations. The vertical dotted line marks generation 2200 in all three plots.** ***(d,e,f)*** **Same as above, but but for a randomly generated Sherrington Kirkpatrick Spin Glasses System over 1000 spins (see main text for details)**

by implicit parallelism differs somewhat from what Holland meant. It is not the average fitness of coarse schemata that gets evaluated and acted upon in parallel, but the effects of vast numbers of coarse schema *partitions*. Significantly, the defining length of the schemata in these partitions need not be low. The implicit parallelism described in this paper is thus of a more powerful kind than that described by Holland. Readers seeking additional evidence of implicit parallelism in UGAs are referred to Chapter 3 of [5].

A second corollary is that the idea of a *hyperscape* is much more helpful than the idea of a landscape [30, 17] for understanding UGA behavior. Landscapes and hyperscapes are both just ways of geometrically conceptualizing fitness functions. Landscapes draw one's attention to the interplay between fitness and neighborhood structure, whereas hyperscapes focus one's attention on the *statistical* fitness properties of individual hyperplanes, and the spatial relationships between hyperplanes—lower order hyperplanes can *contain* higher order hyperplanes, hyperplanes can *intersect* each other, and disjoint hyperplanes belonging to the same hyperplane partition can be regarded as *parallel*. The use of hyperscapes for understanding GA dynamics originated with Holland [12] and was popularized by Goldberg [10]. Unfortunately, the use of hyperscapes tends to be associated with belief in the building block hypothesis. With the building block hypothesis falling into disrepute [9, 25], hyperscapes no longer enjoy the level of attention and interest they once did. The hyperclimbing hypothesis resurrects the hyperscape as a legitimate object of study, and posits that a

hyperscape feature called *staggered conditional effects* is the key to understanding the UGA's capacity for optimization.

We see this paper as a foray into a new and exciting area of research. Much work remains:

- The hyperclimbing hypothesis needs to be fleshed out. Understanding the roles played by mutation and crossover in the implementation of hyperclimbing and understanding when a UGA will be deceived by a hyperscape are important goals.
- Predicting unexpected phenomena and validating them should be an ongoing activity. In the interest of making progress, scientists sacrifice certainty, and strike a bargain in which doubt can only be diminished, never eliminated. "Eternal vigilance" [26], in other words, becomes the cost of progress. This means that the work of the scientist, unlike that of the mathematician, is never quite done.
- Useful as it may be as an explanation for optimization in UGAs, the ultimate value of the hyperclimbing hypothesis lies in its *generalizability*. In a previous work [5], the notion of a unit of inheritance—i.e., a gene—was used to generalize this hypothesis to account for optimization in genetic algorithms with strong linkage between chromosomal loci (including genetic algorithms that do not use crossover). It may be possible for the hyperclimbing hypothesis to be generalized further to account for optimization in other kinds of evolutionary algorithms. whose search spaces consist of real valued vectors, trees, graphs, and instances of other data structures, as well as evolutionary algorithms that use complex variation operators (i.e. probabilistic model building genetic algorithms).
- The field's inability to identify a computation of some kind that evolutionary algorithms perform efficiently is a big reason why Evolutionary Computation remains a niche area within Artificial Intelligence. The realization that implicit parallelism is real has the potential to address this shortcoming. The field of Machine Learning, in particular, stands to benefit from advances in EC. Most machine learning problems reduce to optimization problems, so a new appreciation of how large-scale, general-purpose global optimization can be efficiently implemented should be of interest to researchers in this field. Reaching out to this and other sub-communities in ways that resonate is a priority.
- Last and most importantly, the numerous implications of the hyperclimbing hypothesis for the construction of more effective representations and evolutionary algorithms needs to be explored. The simplicity of the hyperclimbing hypothesis has us particularly excited. Staggered conditional effects and implicit parallelism are easy concepts to grasp, and offer a rich set of avenues to explore (branching and backtracking in hyperspace are two immediate ideas). We are curious to see what folks come up with.

The online appendix is available at http://bit.ly/QFHNAk

7. REFERENCES

[1] D.H. Ackley. *A connectionist machine for genetic hillclimbing*. Kluwer Academic Publishers, 1987.

[2] James E. Baker. Adaptive selection methods for genetic algorithms. In John J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*. Lawrence Erlbaum Associates, Publishers, 1985.

[3] Alfredo Braunstein, Marc Mézard, and Riccardo Zecchina. Survey propagation: an algorithm for satisfiability. *CoRR*, cs.CC/0212002, 2002.

[4] Keki Burjorjee. Sufficient conditions for coarse-graining evolutionary dynamics. In *Foundations of Genetic Algorithms 9 (FOGA IX)*, 2007.

[5] K.M. Burjorjee. *Generative Fixation: A Unifed Explanation for the Adaptive Capacity of Simple Recombinative Genetic Algorithms*. PhD thesis, Brandeis University, 2009.

[6] T. H. Cormen, C. H. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

[7] Kenneth A De Jong and William M Spears. A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 5(1):1–26, 1992.

[8] L.J. Eshelman, R.A. Caruana, and J.D. Schaffer. Biases in the crossover landscape. *Proceedings of the third international conference on Genetic algorithms table of contents*, pages 10–19, 1989.

[9] D. B. Fogel. *Evolutionary Computation : Towards a New Philosophy of Machine Intelligence*. IEEE press, 2006.

[10] David E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[11] David E. Goldberg. *The Design Of Innovation*. Kluwer Academic Publishers, 2002.

[12] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1975.

[13] John H. Holland. Building blocks, cohort genetic algorithms, and hyperplane-defined functions. *Evolutionary Computation*, 8(4):373–391, 2000.

[14] Holger H. Hoos and Thomas Stützle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2004.

[15] Li Huifang and Li Mo. A new method of image compression based on quantum neural network. In *International Conference of Information Science and Management Engineering*, pages p567 – 570, 2010.

[16] E.T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, 2007.

[17] S.A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Biophysical Soc, 1993.

[18] L. Kroc, A. Sabharwal, and B. Selman. Message-passing and local heuristics as decimation strategies for satisfiability. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1408–1414. ACM, 2009.

[19] J. T. Langton, A. A. Prinz, and T. J. Hickey. Combining pixelization and dimensional stacking. In *Advances in Visual Computing*, pages II: 617–626, 2006.

[20] M. Mézard, G. Parisi, and R. Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, 2002.

[21] Melanie Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, Cambridge, MA, 1996.

[22] A.E. Nix and M.D. Vose. Modeling genetic algorithms with Markov chains. *Annals of Mathematics and Artificial Intelligence*, 5(1):79–88, 1992.

[23] Martin Pelikan. Finding ground states of sherrington-kirkpatrick spin glasses with hierarchical boa and genetic algorithms. In *GECCO 2008: Proceedings of the 10th annual conference on Genetic and Evolutionary Computation Conference*, 2008.

[24] Karl Popper. *The Logic Of Scientific Discovery*. Routledge, 2007.

[25] C.R. Reeves and J.E. Rowe. *Genetic Algorithms: Principles and Perspectives: a Guide to GA Theory*. Kluwer Academic Publishers, 2003.

[26] Alexander Rosenbluth and Norbert Wiener. Purposeful and non-purposeful behavior. *Philosophy of Science*, 18, 1951.

[27] B. Selman, H. Kautz, and B. Cohen. Local search strategies for satisfiability testing. *Cliques, coloring, and satisfiability: Second DIMACS implementation challenge*, 26:521–532, 1993.

[28] William M. Spears and Kenneth De Jong. On the virtues of parameterized uniform crossover. In R. K. Belew and L. B. Booker, editors, *Proc. of the Fourth Int. Conf. on Genetic Algorithms*, pages 230–236, San Mateo, CA, 1991. Morgan Kaufmann.

[29] G. Syswerda. Uniform crossover in genetic algorithms. In J. D. Schaffer, editor, *Proceeding of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann, 1989.

[30] Sewall Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. In *Proceedings of the Sixth Annual Congress of Genetics*, 1932.

APPENDIX

A. THE HYPERCLIMBING HEURISTIC: FORMAL DESCRIPTION

Introducing new terminology and notation where necessary, we present a formal description of the hyperclimbing heuristic. For any positive integer ℓ, let $[\ell]$ denote the set $\{1, \ldots, \ell\}$, and let $\mathfrak{B}_\ell$ denote the set of all binary strings of length ℓ. For any binary string g, let g_i denote the i^{th} bit of g. For any set X, let $\mathbb{P}^{\mathbb{X}}$ denote the power set of X. Let $\mathbb{S}_\ell$ and $\mathbb{SP}_\ell$ denote the set of all schemata and schema partitions, respectively, of the set $\mathfrak{B}_\ell$. We define the *schema model set of* ℓ, denoted $\mathbb{SM}_\ell$, to be the set $\{h : D \to \{0,1\} | D \in \mathbb{P}^{[l]}\}$. Each each member of this set is a mapping from the defining bits of a schema to their values.

Given some schema $\gamma \subset \mathfrak{B}_\ell$, let $\pi(\gamma)$ denote the set $\{i \in [\ell] \,|\, \forall x, y \in \gamma, x_i = y_i\}$. We define a *schema modeling function* $\mathbf{SMF}_\ell : \mathbb{S}_\ell \to \mathbb{SM}_\ell$ as follows: for any $\gamma \in \mathbb{S}_\ell$, $\mathbf{SMF}_\ell$ maps γ to the function $h : \pi(\gamma) \to \{0,1\}$ such that for any $g \in \gamma$ and any $i \in \pi(\gamma)$, $h(i) = g_i$. We define a *schema partition modeling function* $\mathbf{SPMF}_\ell : \mathbb{SP}_\ell \to \mathbb{P}^{[\ell]}$ as follows: for any $\Gamma \in \mathbb{SP}_\ell$, $\mathbf{SPMF}_\ell(\Gamma) = \pi(\gamma)$, where $\gamma \in \Gamma$. As $\pi(\psi) = \pi(\xi)$ for all $\psi, \xi \in \Gamma$, the schema partition modeling function is well defined. It is easily seen that $\mathbf{SPF}_\ell$ and $\mathbf{SPMF}_\ell$ are both bijective. For any schema model $h \in \mathbb{SM}_\ell$, we denote $\mathbf{SMF}_\ell^{-1}(h)$ by $[\![h]\!]_\ell$. Likewise, for any "schema partition model" $S \in \mathbb{P}^{[\ell]}$ we denote $\mathbf{SPMF}_\ell^{-1}(S)$ by $[\![S]\!]_\ell$. Going in the forward direction, for any schema $\gamma \in \mathbb{S}_\ell$, we denote $\mathbf{SMF}_\ell(\gamma)$ by $\langle\gamma\rangle$. Likewise, for any schema partition $\Gamma \in \mathbb{SP}_\ell$, we denote $\mathbf{SPMF}_\ell(\Gamma)$ by $\langle\Gamma\rangle$. We drop the ℓ when going in this direction, because its value in each case is ascertainable from the operand. For any schema partition Γ, and any schema $\gamma \in \Gamma$, the *order* of Γ, and the *order* of γ is $|\langle\Gamma\rangle|$.

For any two schema partitions $\Gamma_1, \Gamma_2 \in \mathbb{SP}_\ell$, we say that Γ_1 and Γ_2 are *orthogonal* if the models of Γ_1 and Γ_2 are disjoint (i.e., $\langle\Gamma_1\rangle \cap \langle\Gamma_2\rangle = \emptyset$). Let Γ_1 and Γ_2 be orthogonal schema partitions in $\mathbb{SP}_\ell$, and let $\gamma_1 \in \Gamma_1$ and $\gamma_2 \in \Gamma_2$ be two schemata. Then the concatenation $\Gamma_1\Gamma_2$ denotes the schema partition $[\![\langle\Gamma_1\rangle \cup \langle\Gamma_2\rangle]\!]_\ell$, and the concatenation $\gamma_1\gamma_2$ denotes the schema $[\![h : \langle\Gamma_1\rangle \cup \langle\Gamma_2\rangle \to \{0,1\}]\!]_\ell$ such that for any $i \in \langle\Gamma_1\rangle$, $h(i) = \langle\gamma_1\rangle(i)$, and for any $i \in \langle\Gamma_2\rangle$, $h(i) = \langle\gamma_2\rangle(i)$. Since $\langle\Gamma_1\rangle$ and $\langle\Gamma_2\rangle$ are disjoint, $\gamma_1\gamma_2$ is well defined. Let Γ_1 and Γ_2 be orthogonal schema partitions, and let $\gamma_1 \in \Gamma_1$ be some schema. Then $\gamma.\Gamma_2$ denotes the set $\{\gamma\xi \in \Gamma_1\Gamma_2 \,|\, \xi \in \Gamma_2\}$.

Given some (possibly stochastic) fitness function f over the set $\mathfrak{B}_\ell$, and some schema $\gamma \in \mathbb{S}_\ell$, we define the fitness of γ, denoted $F_\gamma^{(f)}$, to be a random variable that gives the fitness value of a binary string drawn from the uniform distribution over γ. For any schema partition $\Gamma \in \mathbb{SP}_\ell$, we define the *effect* of Γ, denoted $\mathbf{Effect}[\Gamma]$, to be the variance[7] of the expected fitness values of the schemata of Γ. In other words,

$$\mathbf{Effect}[\Gamma] = 2^{-|\langle\Gamma\rangle|} \sum_{\gamma \in \Gamma} \left(\mathbf{E}[F_\gamma^{(f)}] - 2^{-|\langle\Gamma\rangle|} \sum_{\xi \in \Gamma} \mathbf{E}[F_\xi^{(f)}] \right)^2$$

Let $\Gamma_1, \Gamma_2 \in \mathbb{SP}_\ell$ be schema partitions such that $\langle\Gamma_1\rangle \subset \langle\Gamma_2\rangle$. It is easily seen that $\mathbf{Effect}[\Gamma_1] \leq \mathbf{Effect}[\Gamma_2]$. With equality if and only if $F_{\gamma_2}^{(f)} = F_{\gamma_1}^{(f)}$ for all schemata $\gamma_1 \in \Gamma_1$ and $\gamma_2 \in \Gamma_2$ such that $\gamma_2 \subset \gamma_1$. This condition is unlikely to arise in practice; therefore, for all practical purposes, the effect of a given schema partition decreases as the partition becomes coarser. The schema partition $[\![\,[l]\,]\!]_\ell$ has the maximum effect. Let Γ and Ψ be two orthogonal schema partitions, and let $\gamma \in \Gamma$ be some schema . We define the conditional effect of Ψ *given* γ, denoted $\mathbf{Effect}[\Psi|\gamma]$, as follows:

$$\mathbf{Effect}[\Psi|\gamma] = 2^{-|\langle\Psi\rangle|} \sum_{\psi \in \Psi} \left(\mathbf{E}[F_{\gamma\psi}^{(f)}] - 2^{-|\langle\Psi\rangle|} \sum_{\xi \in \Psi} \mathbf{E}[F_{\gamma\xi}^{(f)}] \right)^2$$

A hyperclimbing heuristic works by evaluating the fitness of samples drawn initially from the uniform distribution over the search space. It finds a coarse schema partition Γ with a non-zero effect, and limits future sampling to some schema γ of this partition whose average sampling fitness is greater than the mean of the average sampling fitness values of the schemata in Γ. By limiting future sampling in this way, the heuristic raises the expected fitness of all future samples. The heuristic limits future sampling to some schema

[7]We use variance because it is a well known measure of dispersion. Other measures of dispersion may well be substituted here without affecting the discussion

by fixing the defining bits [21] of that schema in all future samples. The unfixed loci constitute a new (smaller) search space to which the hyperclimbing heuristic is then recursively applied. Crucially, coarse schema partitions orthogonal to Γ that have undetectable *unconditional* effects, may have detectable effects when conditioned by γ.

B. VISUALIZING STAIRCASE FUNCTIONS

The following addressing scheme allows us to project a high dimensional fitness function onto a two dimensional plot.

DEFINITION 2. *A* refractal addressing system *is a tuple* (m, n, X, Y)*, where* m *and* n *are positive integers and* X *and* Y *are matrices with* m *rows and* n *columns such that the elements in* X *and* Y *are distinct positive integers from the set* $[2mn]$*, such that for any* $k \in [2mn]$*,* k *is in* $X \iff k$ *is not in* Y *(i.e. the elements of* $[2mn]$ *are evenly split between* X *and* Y*).*

A refractal addressing system (m, o, X, Y) determines how the set $\mathfrak{B}_{2mn}$ gets mapped onto a $2^{mn} \times 2^{mn}$ grid of pixels. For any bitstring $g \in \mathfrak{B}_{2mn}$ the xy-address (a tuple of two values, each between 1 and 2^{mn}) of the pixel representing g is given by Algorithm 3.

Example: Let $(h = 4, o = 2, \delta = 3, \ell = 16, L, V)$ be the descriptor of a staircase function f, such that

$$V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

Let $A = (m = 4, n = 2, X, Y)$ be a refractal addressing system such that $X_{1:} = L_{1:}$, $Y_{1:} = L_{2:}$, $X_{2:} = L_{3:}$, and $Y_{2:} = L_{4:}$. A *refractal plot*[8] of f is shown in Figure 4a.

This image was generated by querying f with all 2^{16} elements of $\mathfrak{B}_{16}$, and plotting the fitness value of each bitstring as a greyscale pixel at the bitstring's refractal address under the addressing system A. The fitness values returned by f have been scaled to use the full range of possible greyscale shades.[9] Lighter shades signify greater fitness. The four stages of f can easily be discerned.

Suppose we generate another refractal plot of f using the same addressing system A, but a different random number generator seed; because f is stochastic, the greyscale value of any pixel in the resulting plot will then most likely differ from that of its homolog in the plot shown in Figure 4a. Nevertheless, our ability to discern the stages of f would not be affected. In the same vein, note that when specifying A, we have not specified the values of the last two rows of X and Y; given the definition of f it is easily seen that these values are immaterial to the discernment of its "staircase structure".

On the other hand, the values of the first two rows of X and Y are highly relevant to the discernment of this structure. Figure 4b shows a refractal plot of f that was obtained using a refractal addressing system $A' = (m = 4, n = 2, X', Y')$ such that $X'_{4:} = L_{1:}$, $Y'_{4:} = L_{2:}$, $X'_{3:} = L_{3:}$, and $Y'_{3:} = L_{4:}$. Nothing remotely resembling a staircase is visible in this plot.

[8]The term "refractal plot" describes the images that result when *dimensional stacking* is combined with *pixelation* [19].

[9]We used the Matlab function `imagesc()`

Algorithm 3: The algorithm for determining the (x, y)-address of a chromosome under the refractal addressing system (m, n, X, Y). The function BIN-TO-INT returns the integer value of a binary string.

Input: g is a chromosome of length $2mn$

```
granularity ← 2^{mn}/2^n
x ← 1
y ← 1
for i ← 1 to m do
    x ← x + granularity * BIN-TO-INT (Ξ_{X_{i:}}(g))
    y ← y + granularity * BIN-TO-INT (Ξ_{Y_{i:}}(g))
    granularity ← granularity/2^n
end
return x, y
```

The lesson here is that the discernment of the fitness staircase inherent within a staircase function depends critically on how one 'looks' at this function. In determining the 'right' way to look at f we have used information about the descriptor of f, specifically the values of h, o, and L. This information will not be available to an algorithm which only has query access to f.

Even if one knows the right way to look at a staircase function, the discernment of the fitness staircase inherent within this function can still be made difficult by a low value of the increment parameter. Figure 5 lets us visualize the decrease in the salience of the fitness staircase of f that accompanies a decrease in the increment parameter of this staircase function. In general, a decrease in the increment results in a decrease in the 'contrast' between the stages of that function, and an increase the amount of computation required to discern these stages.

C. ANALYSIS OF STAIRCASE FUNCTIONS

Let ℓ be some positive integer. Given some (possibly stochastic) fitness function f over the set $\mathfrak{B}_\ell$, and some schema $\gamma \subseteq \mathfrak{B}_\ell$ we define the *fitness signal* of γ, denoted $S(\gamma)$, to be $\mathbf{E}[F_\gamma^{(f)}] - \mathbf{E}[F_{\mathfrak{B}_\ell}^{(f)}]$. Let $\gamma_1 \subseteq \mathfrak{B}_\ell$ and $\gamma_2 \subseteq \mathfrak{B}_\ell$ be schemata in two orthogonal schema partitions. We define the *conditional fitness signal of* γ_1 *given* γ_2, denoted $S(\gamma_1 \,|\, \gamma_2)$, to be the difference between the fitness signal of $\gamma_1\gamma_2$ and the fitness signal of γ_2, i.e. $S(\gamma_1 \,|\, \gamma_2) = S(\gamma_1\gamma_2) - S(\gamma_2)$. Given some staircase function f we denote the i^{th} step of f by $\lfloor f \rfloor_i$ and denote the i^{th} stage of f by $\lceil f \rceil_i$.

Let f be a staircase function with descriptor $(h, o, \delta, \ell, L, V)$. For any integer $i \in [h]$, the fitness signal of $\lfloor f \rfloor_i$ is one measure of the difficulty of "directly" identifying step i (i.e., the difficulty of determining step i without first determining any of the preceding steps $1, \ldots, i-1$). Likewise, for any integers i, j in $[h]$ such that $i > j$, the conditional fitness signal of step i given stage j is one measure of the difficulty of directly identifying step i given stage j (i.e. the difficulty of determining $\lfloor f \rfloor_i$ given $\lceil f \rceil_j$ without first determining any of the intermediate steps $\lfloor f \rfloor_{j+1}, \ldots, \lfloor f \rfloor_{i-1}$).

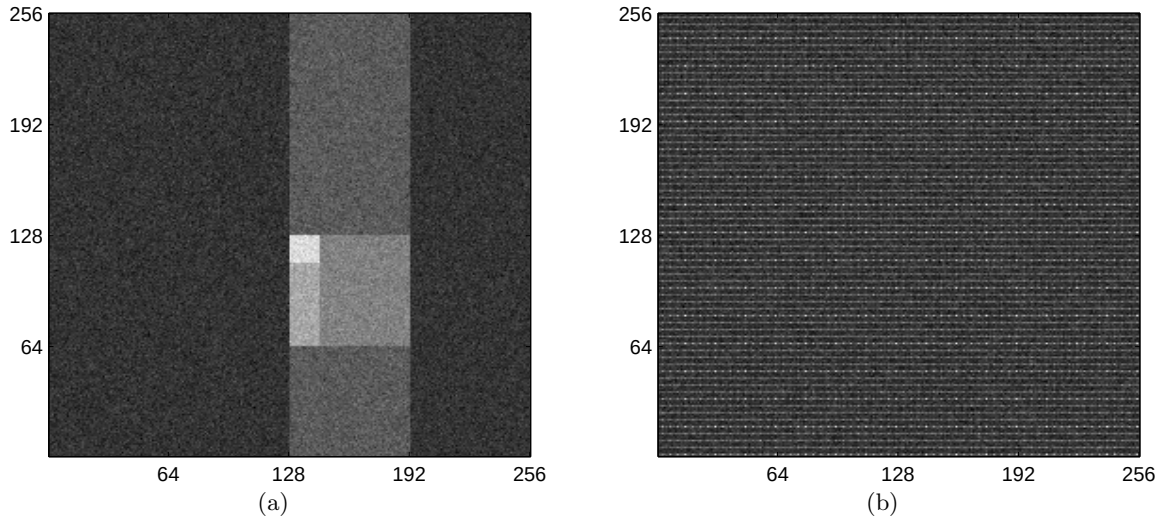


Figure 4: A refractal plot of the staircase function f under the refractal addressing systems A (*left*) and A' (*right*).

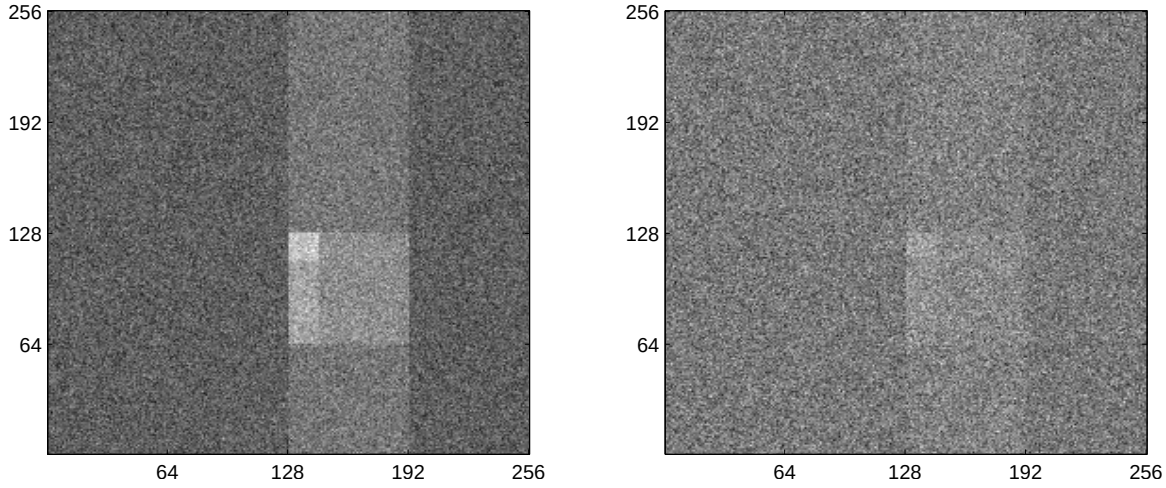


Figure 5: Refractal plots under A of two staircase functions, which differ from f only in their increments—1 *(left plot)* and 0.3 *(right plot)* as opposed to 3.

By Theorem 1 (Appendix C), for any $i \in [h]$, the unconditional fitness signal of step i is

$$\frac{\delta}{2^{o(i-1)}}$$

This value decreases exponentially with i and o. It is reasonable, therefore, to suspect that the direct identification of step i of f quickly becomes infeasible with increases in i and o. Consider, however, that by Corollary 1, for any $i \in \{2, \ldots, h\}$, the *conditional* fitness signal of step i *given* stage $(i-1)$ is δ, a *constant* with respect to i. Therefore, if some algorithm can identify the first step of f, one should be able to use it to iteratively identify all remaining steps in time and fitness queries that scale linearly with the height of f.

LEMMA 1. *For any staircase function f with descriptor $(h, o, \delta, \ell, L, V)$, and any integer $i \in [h]$, the fitness signal of stage i is $i\delta$.*

PROOF: Let x be the expected fitness of $\mathfrak{B}_\ell$ under uniform sampling. We first prove the following claim:

CLAIM 1. *The fitness signal of stage i is $i\delta - x$*

The proof of the claim follows by induction on i. The base case, when $i = h$ is easily seen to be true from the definition of a staircase function. For any $k \in \{2, \ldots, h\}$, we assume that the hypothesis holds for $i = k$, and prove that it holds for $i = k-1$. For any $j \in [h]$, let $\Gamma_j \in \mathbb{SP}_\ell$ denote the schema partition containing step i. The fitness signal of stage $k-1$

is given by

$$\frac{1}{2^o}\left(S(\lceil f\rceil_k)+\sum_{\psi\in\Gamma_k\setminus\{\lfloor f\rfloor_k\}}S(\lceil f\rceil_{k-1}\psi)\right)$$

$$=\frac{k\delta-x}{2^o}+\frac{2^o-1}{2^o}\left(\delta(k-1)-\frac{\delta}{2^o-1}-x\right)$$

where the first term of the right hand side of the equation follows from the inductive hypothesis, and the second term follows from the definition of a staircase function. Manipulation of this expression yields

$$\frac{k\delta+(2^o-1)\delta(k-1)-\delta-2^ox}{2^o}$$

which, upon further manipulation, yields $(k-1)\delta-x$.

This completes the proof of the claim. To prove the lemma, we must prove that x is zero. By claim 1, the fitness signal of the first stage is $\delta-x$. By the definition of a staircase function then,

$$x=\frac{\delta-x}{2^o}+\frac{2^o-1}{2^o}\left(-\frac{\delta}{2^o-1}\right)$$

Which reduces to

$$x=-\frac{x}{2^o}$$

Clearly, x is zero. □

COROLLARY 1. *For any $i\in\{2,\ldots,h\}$, the conditional fitness signal of step i given stage $i-1$ is δ*

PROOF The conditional fitness signal of step i given stage $i-1$ is given by

$$\begin{aligned}&S(\lfloor f\rfloor_i\mid\lceil f\rceil_{i-1})\\&=S(\lceil f\rceil_i)-S(\lceil f\rceil_{i-1})\\&=(i\delta-(i-1)\delta)\\&=\delta\ \square\end{aligned}$$

THEOREM 1. *For any staircase function f with descriptor $(h,o,\delta,\sigma,\ell,L,V)$, and any integer $i\in[h]$, the fitness signal of step i is $\delta/2^{o(i-1)}$.*

PROOF: For any $j\in[h]$, let $\Lambda_j\in\mathbb{SP}_\ell$ denote of the partition containing stage j, and let $\Gamma_j\in\mathbb{SP}_\ell$ denote of the partition containing step j. We first prove the following claim

CLAIM 2. *For any $i\in[h]$,*

$$\sum_{\xi\in\Lambda i\setminus\{\lceil f\rceil_i\}}S(\xi)=-i\delta$$

The proof of the claim follows by induction on i. The proof for the base case ($i=1$) is as follows:

$$\sum_{\xi\in\Lambda_1\setminus\{\lceil f\rceil_1\}}S(\xi)=(2^o-1)\left(\frac{-\delta}{2^o-1}\right)=-\delta$$

For any $k\in[h-1]$ we assume that the hypothesis holds for $i=k$, and prove that it holds for $i=k+1$.

$$\sum_{\xi\in\Lambda_{k+1}\setminus\{\lceil f\rceil_{k+1}\}}S(\xi)$$

$$=\sum_{\psi\in\Gamma_{k+1}\setminus\{\lfloor f\rfloor_{k+1}\}}S(\lceil f\rceil_k\psi)+\sum_{\xi\in\Lambda_k\setminus\{\lceil f\rceil_k\}}\ \sum_{\psi\in\Gamma_{k+1}}S(\xi\psi)$$

$$=\sum_{\psi\in\Gamma_{k+1}\setminus\{\lceil f\rceil_{k+1}\}}S(\lceil f\rceil_k\psi)+\sum_{\psi\in\Gamma_{k+1}}\ \sum_{\xi\in\Lambda_k\setminus\{\lceil f\rceil_k\}}S(\xi\psi)$$

$$=(2^o-1)S(\lceil f\rceil_k)+2^o\left(\sum_{\xi\in\Lambda_k\setminus\{\lceil f\rceil_k\}}S(\xi)\right)$$

where the first and last equalities follow from the definition of a staircase function. Using Lemma 1 and the inductive hypothesis, the right hand side of this expression can be seen to equal

$$(2^o-1)\left(k\delta-\frac{\delta}{2^o-1}\right)-2^ok\delta$$

which, upon manipulation, yields $-\delta(k+1)$.

For a proof of the theorem, observe that step 1 and stage 1 are the same schema. So, by Lemma 1, $S(\lfloor f\rfloor_1)=\delta$. Thus, the theorem holds for $i=1$. For any $i\in\{2,\ldots,h\}$,

$$\begin{aligned}S(\lfloor f\rfloor_i)&=\frac{1}{(2^o)^{i-1}}\left(S(\lceil f\rceil_i)+\sum_{\xi\in\Lambda_{i-1}\setminus\{\lceil f\rceil_{i-1}\}}S(\xi\lfloor f\rfloor_k)\right)\\&=\frac{1}{(2^o)^{i-1}}\left(S(\lceil f\rceil_i)+\sum_{\xi\in\Lambda_{i-1}\setminus\{\lceil f\rceil_{i-1}\}}S(\xi)\right)\end{aligned}$$

where the last equality follows from the definition of a staircase function. Using Lemma 1 and Claim 2, the right hand side of this equality can be seen to equal

$$\frac{i\delta-(i-1)\delta}{(2^o)^{i-1}}$$

$$=\frac{\delta}{2^{o(i-1)}}\quad\square$$