# Optimization by Decimation: Explaining Adaptation in Genetic Algorithms With Uniform Crossover

**Keki M. Burjorjee** KEKIB@CS.BRANDEIS.EDU
*Amazon.com, 1200 12th Ave. South, # 1200,*
*Seattle, WA 98144, USA*

## Abstract

We submit the *hyperclimbing hypothesis*—an explanation for adaptation in genetic algorithms with uniform crossover (UGAs). Hyperclimbing is a stochastic search heuristic that works by *decimating* a search space, i.e. by iteratively fixing the values of small numbers of search space attributes. Global decimation is known to be an effective way to approach large instances of hard constraint satisfaction problems. The hyperclimbing hypothesis holds that UGAs work by implicitly implementing *efficient* global decimation. Proof of concept for this hypothesis comes from the use of a novel analytic technique involving the exploitation of algorithmic symmetry. We also present experimental results that show that a simple tweak inspired by the hyperclimbing hypothesis significantly improves the performance of a UGA on an instance of Uniform Random MAX-3SAT . The hyperclimbing hypothesis suggests that other kinds of evolutionary algorithms may also work by implicitly implementing efficient global decimation.

## 1 Introduction

Over several decades of use in diverse scientific and engineering fields, evolutionary optimization has acquired a reputation for being a kind of universal acid—a general purpose approach that routinely procures useful solutions to optimization problems with rugged, dynamic, and stochastic cost functions over search spaces consisting of strings, vectors, trees, and instances of other kinds of data structures (Fogel, 2006). Remarkably, despite years of research, the means by which evolutionary algorithms work is still the subject of much debate. An abiding mystery of the field is the widely observed utility of genetic algorithms with uniform crossover (Syswerda, 1989; Rudnick et al., 1994; Pelikan et al., 2008; Huifang and Mo, 2010). The use of uniform crossover (Ackley, 1987; Syswerda, 1989) in genetic algorithms causes genetic loci to be unlinked, i.e. recombine freely. It is generally acknowledged that the adaptive capacity of genetic algorithms with this kind of crossover cannot be explained within the rubric of the *building block hypothesis*—currently the reigning explanation for adaptation in genetic algorithms with strong linkage between loci. Yet, no alternate explanation for adaptation in genetic algorithms with uniform crossover (UGAs) has gained currency. We submit the *hyperclimbing hypothesis*, a new explanation for adaptation in UGAs. This hypothesis holds that UGAs perform adaptation by implicitly and efficiently implementing a global decimation strategy called *hyperclimbing*.

Global decimation is currently the state of the art approach to solving large, hard instances of random SAT problems (Kroc et al., 2009). Conventional global decimation strategies use message passing algorithms—e.g. Survey Propagation (Mézard et al., 2002), Belief Propagation, Warning Propagation (Braunstein et al., 2002)—to gather statistical information about the space being searched. This information is used to fix the values of one, or a small number, of the search space attributes, effectively reducing the size of the search space. The decimation strategy is then recursively applied to the smaller search space. And so on. Survey Propagation, perhaps the best known global decimation strategy, has been used along with *Walksat* (Selman et al., 1996) to solve million variable instances of Random SAT problems close to the SAT/UNSAT threshold. The hyperclimbing hypothesis holds that in practice, UGAs perform adaptation by efficiently decimating the search spaces to which they are applied. Unlike conventional decimation strategies, however, UGAs gather global statistical information *implicitly*, by means not involving message passing.

## 2 The Hyperclimbing Heuristic

For a sketch of the workings of a hyperclimbing heuristic, consider a search space $S = \{0,1\}^\ell$, and a (possibly stochastic) fitness function[1] that maps points in $S$ to real values. Given some set of indices $\mathcal{I} \subseteq \{1,\ldots,\ell\}$, $\mathcal{I}$ partitions $S$ into $2^{|\mathcal{I}|}$ subsets 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*, and the subsets of a schema partition are called *schemata* (singular schema) (Mitchell, 1996). Clearly, the smaller the value of $|\mathcal{I}|$ the coarser the schema partition. 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. A hyperclimbing heuristic starts by sampling from the uniform distribution over the 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. This subset constitutes a new (smaller) search space of binary strings to which the hyperclimbing heuristic is recursively applied.

Introducing relevant notation and terminology where necessary, we now present a more precise description of the hyperclimbing heuristic. For any positive integer $\ell$, let $[\ell]$ denote the set $\{1,\ldots,\ell\}$, and let $\mathfrak{B}_\ell$ denote the set of all binary strings of length $\ell$. For any binary string $g$, let $g_i$ denote the $i^{th}$ bit of $g$. We define the *schema partition model set of* $\ell$, denoted $\mathbb{SPM}_\ell$, to be the power set of $[\ell]$, and define the *schema model set of* $\ell$, denoted $\mathbb{SM}_\ell$, to be the set $\{h : D \to \{0,1\} | D \in \mathbb{SPM}_\ell\}$. Let $\mathbb{S}_\ell$ and $\mathbb{SP}_\ell$ be the set of all schemata and schema partitions (Mitchell, 1996), respectively, of the set $\mathfrak{B}_\ell$. 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 $\gamma$ 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* $\mathbf{SPMF}_\ell : \mathbb{SP}_\ell \to \mathbb{SPM}_\ell$ 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

1. A fitness function is a cost function with a twist: the goal is, not to minimize fitness, but to maximize it.

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 $S \in \mathbb{SPM}_\ell$ 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 $\ell$ when going in this direction, because its value in each case is ascertainable from the operand. For any schema partition $\Gamma$, and any schema $\gamma \in \Gamma$, the *order* of $\Gamma$, and the *order* of $\gamma$ is $|\langle\Gamma\rangle|$.

For any two schema partitions $\Gamma_1, \Gamma_2 \in \mathbb{SP}_\ell$, we say that $\Gamma_1$ and $\Gamma_2$ are *orthogonal* if the models of $\Gamma_1$ and $\Gamma_2$ are disjoint (i.e., $\langle\Gamma_1\rangle \cap \langle\Gamma_2\rangle = \emptyset$). Let $\Gamma_1$ and $\Gamma_2$ 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 $\Gamma_1$ and $\Gamma_2$ 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 $\gamma$, denoted $F_\gamma^{(f)}$, to be a random variable that gives the fitness value of a binary string drawn from the uniform distribution over $\gamma$. For any schema partition $\Gamma \in \mathbb{SP}_\ell$, we define the *effect* of $\Gamma$, denoted $\mathbf{Effect}[\Gamma]$, to be the variance[2] of the expected fitness values of the schemata of $\Gamma$. 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 $\Gamma$ and $\Psi$ be two orthogonal schema partitions, and let $\gamma \in \Gamma$ be some schema . We define the conditional effect of $\Psi$ *given* $\gamma$, 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 $\Gamma$ with a non-zero effect, and limits future sampling to some schema $\gamma$ of this partition whose average sampling fitness is greater then the mean of the average sampling fitness values of the schemata in $\Gamma$. 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 by fixing the defining bits (Mitchell, 1996) of that schema in all future samples. The

2. 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

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

**Input**: $g$ is a chromosome of length $\ell$

$x \leftarrow$ some value drawn from the distribution $\mathcal{N}(0, 1)$
**for** $i \leftarrow 1$ *to* $h$ **do**
  **if** $\Xi_{L_{i:}}(g) = V_{i1} \dots V_{io}$ **then**
    $x \leftarrow x + \delta$
  **else**
    $x \leftarrow x - (\delta/(2^o - 1))$
    **break**
  **end**
**end**
**return** $x$

unfixed loci constitute a new (smaller) search space to which the hyperclimbing heuristic is then recursively applied. Crucially, coarse schema partitions orthogonal to $\Gamma$ that have undetectable *unconditional* effects, may have detectable effects when conditioned by $\gamma$. And so on.

# 3 Proof of Concept

We introduce a parameterized stochastic fitness function, called a *staircase function*, 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 *optimally* with changes to key parameters. An experimental test validates this conclusion.

**Definition 1** *A staircase function descriptor is a 6-tuple $(h, o, \delta, \ell, L, V)$ where $h$, $o$ and $\ell$ are positive integers such that $ho \leq \ell$, $\delta$ 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 $[\ell]$.*

For any positive integer $\ell$, let $[\ell]$ denote the set $\{1, \dots, \ell\}$, and let $\mathfrak{B}_\ell$ denote the set of binary strings of length $\ell$. 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 $b_1, \dots, b_k$ 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 $\ell$ given by Algorithm 1. We call $h, o, \delta$, and $\ell$ the *height, order, increment* and *span*, respectively, of $f$. For any $i \in [h]$ we define *step* $i$ of $f$ to be the schema $\{g \in \mathfrak{B}_\ell | \Xi_{L_{i:}}(g) = V_{i1} \dots V_{io}\}$, and define *stage* $i$ of $f$ to be the schema $\{g \in \mathfrak{B}_\ell | (\Xi_{L_{1:}}(g) = V_{11} \dots V_{1o}) \wedge \dots \wedge (\Xi_{L_{i:}}(g) = V_{i1} \dots V_{io})\}$.

We say that some step of the staircase function has 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 is computationally expensive to identify higher steps of a staircase function without identifying lower steps first; the difficulty of climbing step $i \in [h]$ *given* stage $i-1$, however, is non-increasing with respect to $i$ (Appendix B).

### 3.1 Visualizing Staircase Functions

The stages of a staircase function can be visualized as a progression of nested *hyperplanes*[3], 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 *3.1*).

**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* $Y$*).*

The 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 2.
**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*[4] of $f$ is shown in Figure 1a.

This image was generated by querying $f$ with every bitstring in $\mathfrak{B}_{16}$, and plotting the resulting fitness value of each chromosome as a greyscale pixel at the chromosome'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[5]. 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 1a. 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".

3. A hyperplane is a geometrical representation of a schema (Goldberg, 1989, p 53).
4. The term "refractal plot" describes the images that result when *dimensional stacking* is combined with *pixelation* (Langton et al., 2006).
5. We used the Matlab function `imagesc()`

**Algorithm 2:** 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 $2mn$

$granularity \leftarrow 2^{mn}/2^n$
$x \leftarrow 1$
$y \leftarrow 1$
**for** $i \leftarrow 1$ *to* $m$ **do**
  $x \leftarrow x + granularity * \text{BIN-TO-INT } (\Xi_{X_{i:}}(g))$
  $y \leftarrow y + granularity * \text{BIN-TO-INT } (\Xi_{Y_{i:}}(g))$
  $granularity \leftarrow granularity/2^n$
**end**
**return** $x, y$

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 1b 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.

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 2 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.

### 3.2 Performance of a UGA on a Staircase Function

Let $f$ be a staircase function with descriptor $(h, o, \delta, \ell, L, V)$, we say that this function 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 basic, then the last three elements of the descriptor of $f$ are fully determinable from the first three; we therefore write this descriptor as $(h, o, \delta)$. 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 $(h, o, \delta)$.

Let $\phi^*$ be the basic staircase function with descriptor $(h = 50, o = 4, \delta = 0.3)$, and let $U$ denote the UGA defined in the materials and methods section (Appendix $A$) with a population size of 500, and a per bit mutation probability of 0.003 (i.e, $p_m = 0.003$). Figure 3a shows that $U$ is capable of robust adaptation when applied to $\phi^*$ (We denote

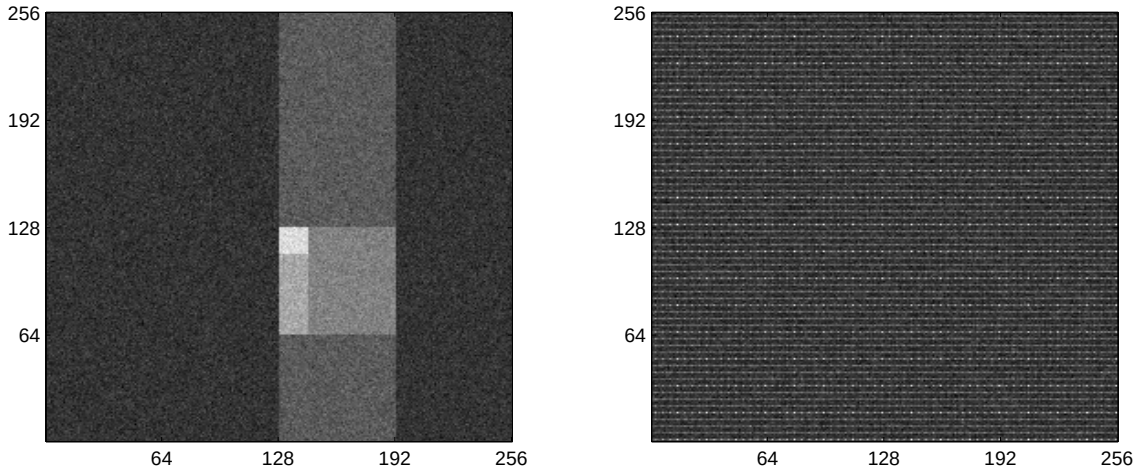



Figure 1: A refractal plot of the staircase function $f$ under the refractal addressing systems $A$ (*left*) and $A'$ (*right*).

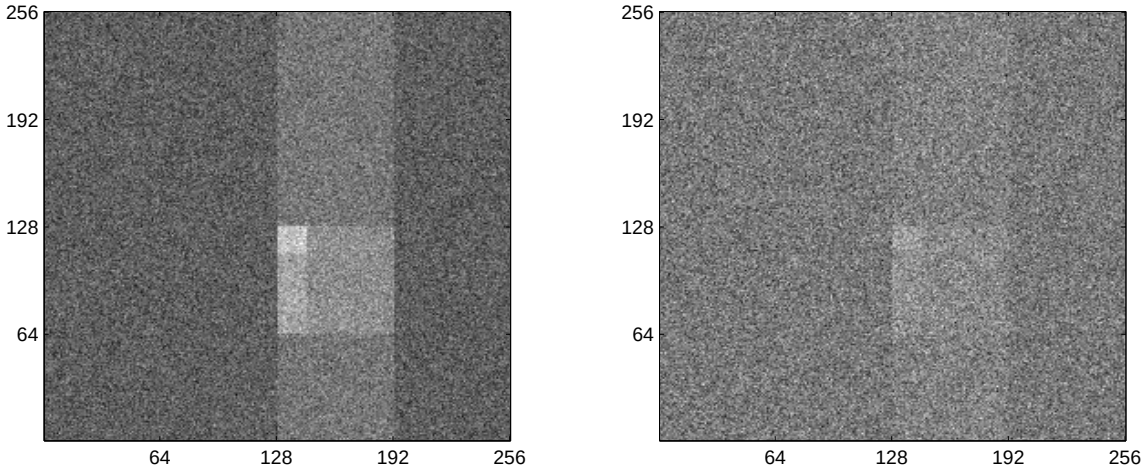



Figure 2: Refractal plots under $A$ of two staircase functions, which differ from $f$ only in their increments—1 *(left plot)* and 0.3 *(right plot)* as opposed to 3.

the resulting algorithm by $U^{\phi^*}$). Figure 3c shows that under the action of $U$, the first four steps of $\phi^*$ go to fixation[6] 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. Figure 1c shows that the UGA does not need to "fully" climb" a step before it

6. We use the terms 'fixation' and 'fixing' loosely. 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.

begins climbing the ones immediately above it. Animation 1 in the online appendix shows that the hyperclimbing behavior of $U^{\phi^*}$ continues beyond the first four steps.

### 3.3 Symmetry Analysis

Formal models of SGAs with finite populations and non-trivial fitness functions (Nix and Vose, 1992), are notoriously unwieldy (Holland, 2000), which is why most theoretical analyses of SGAs assume an infinite population (Liepins and Vose, 1992; Stephens and Waelbroeck, 1999; Wright et al., 2003; Burjorjee, 2007). Unfortunately, since the running time and the number of fitness evaluations required by such models is always infinite, the use of such models precludes the identification of computational efficiencies of the SGA. In the present case, we circumvent the difficulty of formally analyzing finite population SGAs by exploiting some simple symmetries introduced through our definition of staircase functions, and through our use of a crossover operator with no *positional bias*. The absence of positional bias in uniform crossover was highlighted by Eshelman et al.(1989). Essentially, permuting the bits of all strings in each generation using some permutation $\pi$ before crossover, and permuting the bits back using $\pi^{-1}$ after crossover has no effect on the dynamics of a UGA. Another way to elucidate this symmetry is by noting that any homologous crossover operator can be modeled as a string of binary random variables. Only in the case of uniform crossover, however, are these random variables all independent and identically distributed.

A conclusion that follows straightforwardly from the introduced symmetries is as follows: it is easily seen that loci that are not part of any step of a staircase function are immaterial during fitness evaluation. The absence of positional bias in uniform crossover entails that such loci can also be ignored during recombination. Effectively, then, these loci can be "spliced out" without affecting the expected average fitness of the population in any generation. This, and other observations of this type lead us to the conclusions listed below.

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 $x$. Likewise, let $q^{(t)}_{(W^f,i)}(x)$ denote the probability that the frequency of *step* $i$ of $f$ in generation $t$ of $W^f$ is $x$. Finally, for any $x \in \mathbb{R}$, let $r^{(t)}_{W^f}(x)$ denote the probability that the average fitness of the population of $W^f$ in generation $t$ is $x$.

Let $f^*$ be some basic staircase function with descriptor $(h, o, \delta, \sigma)$, and let $f$ be a staircase function with basic form $f^*$. For any UGA $W$, by appreciating the symmetries between the UGAs $W^{f^*}$ and $W^f$ we conclude that

1. For any generation $t$, $r^{(t)}_{W^f} = r^{(t)}_{W^{f^*}}$
2. For any generation $t$, and any $i \in [h]$, $p^{(t)}_{(W^f,i)} = p^{(t)}_{(W^{f^*},i)}$
3. For any generation $t$, and any $i \in [h]$, $q^{(t)}_{(W^f,i)} = q^{(t)}_{(W^{f^*},i)}$

Let $f$ be some staircase function with basic form $\phi^*$. An immediate corollary of conclusions 1 and 3 is that the application of $U$ to $f$ should, discounting deviations due to sampling, produce results identical to those shown in Figures 3a and 3c. We validated this conclusion by applying $U$ to the staircase function $\phi$ with descriptor $(h = 50, o = 4, \delta =$

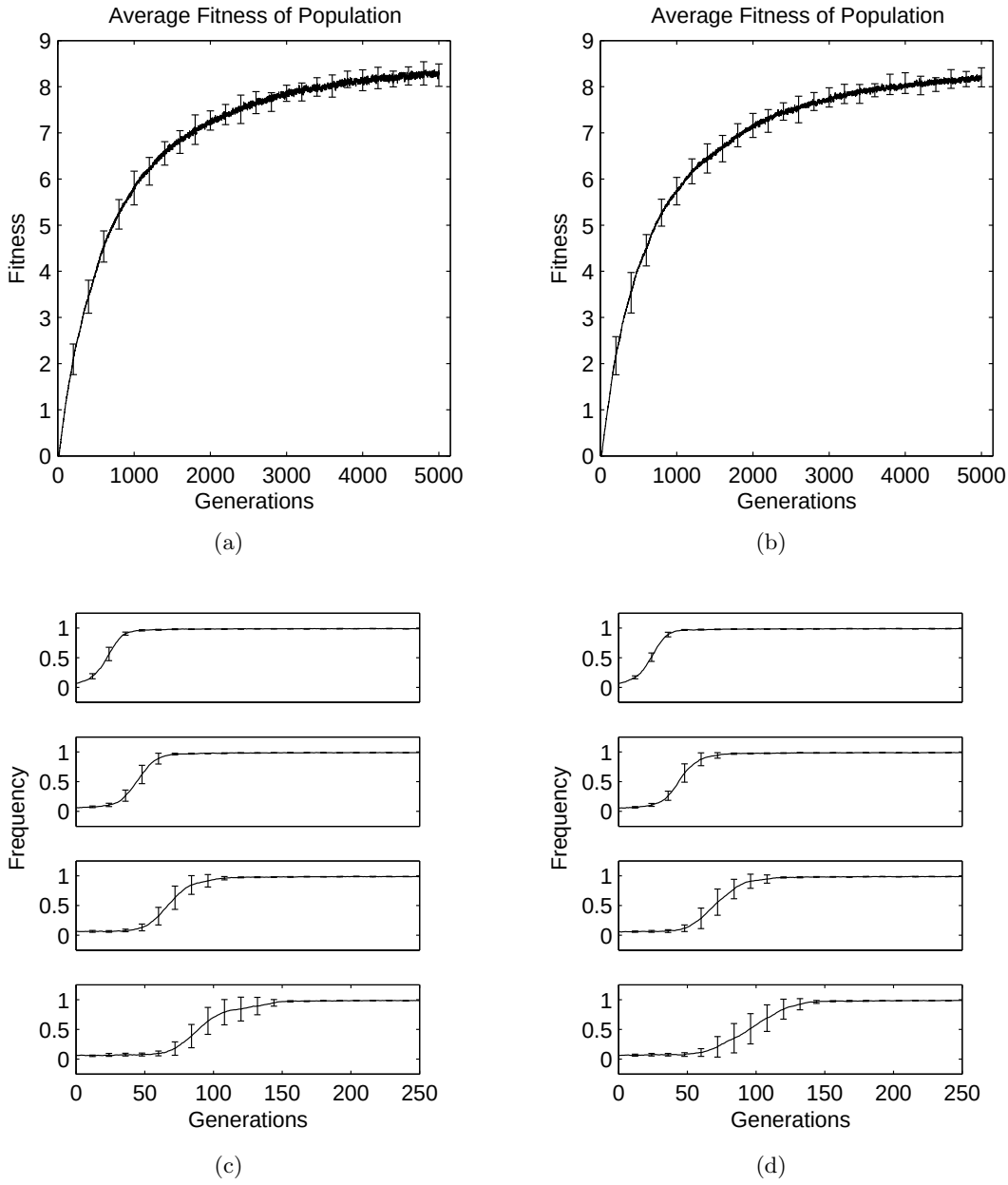



Figure 3: (a) The mean, across 20 trials, of the average fitness of the population of $U^{\phi^*}i$ in each of 5000 generations.The error bars show five standard errors above and below the mean every 200 generations. (b) Going from the top plot to the bottom plot, the mean frequencies, across 20 trials, of the first five steps of the staircase function $U^{\phi^*}$ in each of the first 250 generations. The error bars show three standard errors above and below the mean every 12 generations. (c,d) Same as top left and bottom left, respectively, but for $U^{\phi}$

.

$0.3, \ell = 20000, L, V)$ where $L$ and $V$ were randomly generated. The results are shown in Figures 3b and 3d. 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 changes that appear to be statistically significant.

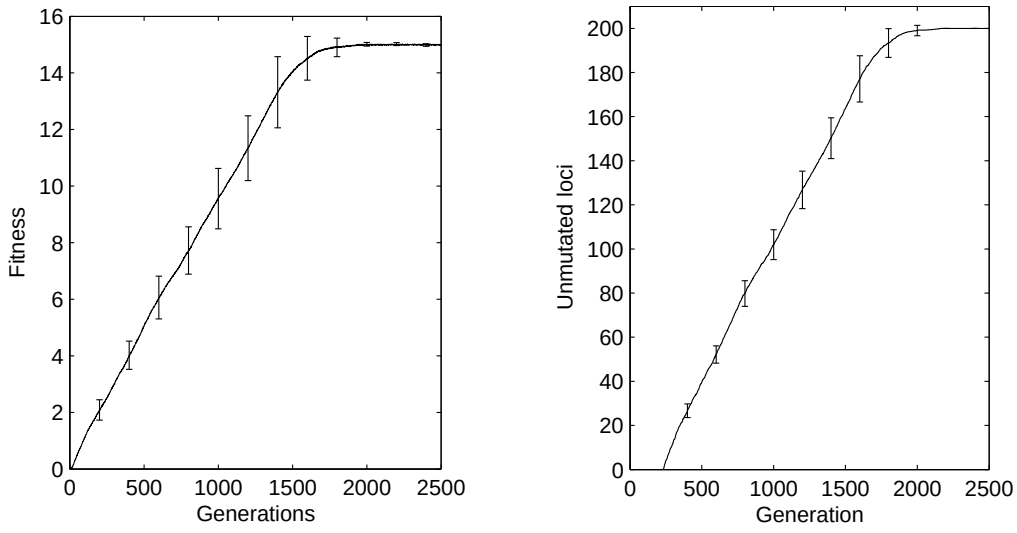
Conclusion 1 entails that a) the number of fitness queries required by a UGA to achieve some fixed increase in the expected average fitness of a population is *constant* with respect to the span of a staircase function, b) the running time required by a UGA scales linearly with the span of a staircase function, and c) that the running time and the number of fitness queries are unaffected by the last two elements in the descriptor of a staircase function.

# 4 Validation

We pause to consider a curious aspect of the behavior of $U^{\phi^*}$. Figure 1 shows that the growth rate of the average fitness of the population of $U^{\phi^*}$ 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 $\phi^*$, why does it fail to identify all remaining steps? To understand why, consider some binary string that belongs to the $i^{th}$ stage of $\phi^*$. 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, $U^{\phi^*}$ is less able to "hold" a population within stage $i$. In light of this observation, we 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 $U^{\phi^*}$. 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 we expect 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 4a. Figure 4b 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 $\phi^*$. Animation 2 in




(a) (b)

Figure 4: *(Left:)* The mean (across 20 trials) of the average fitness of the UGA $U_c$ on the staircase function $\phi^*$. Errorbars show five standard errors above and below the mean every 200 generations. *(Right:)* 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

the online appendix shows the 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.

### 4.1 Application to MAXSAT

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 present the results of an experiment where the use of clamping clearly improves the performance of a UGA on a relatively large instance of MAX-3SAT (Hoos and Stützle, 2004). Let $Q$ denote the UGA defined in the materials and methods section (Appendix $A$) with a population size of 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 (Hoos and Stützle, 2004), 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 5a 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.

We applied $Q$ to *sat* once again; this time, however, we activated clamping in generation 2000. We denote the resulting UGA by $Q_c^{sat}$. 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 5b, and the number of loci that the clamping mechanism left unmutated in each generation is shown in Figure 5c. 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 5b and 5c).

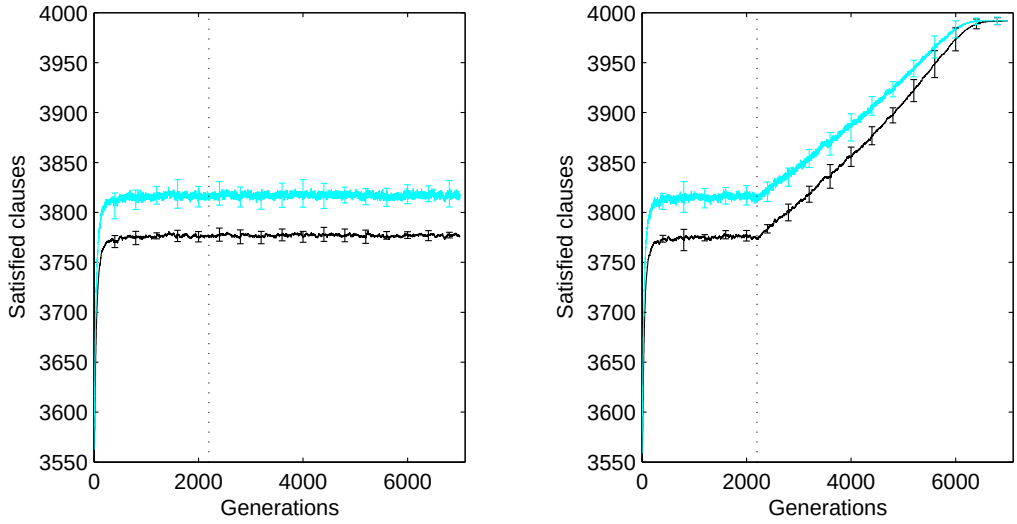
The effectiveness of clamping demonstrated here lends considerable support to the hyperclimbing hypothesis. Further support can be found in the work of Huifang and Mo (Huifang and Mo, 2010) where the use of clamping improved the performance of a UGA on a completely different problem (optimizing the weights of a quantum neural network).

# 5 Conclusion

Simple genetic algorithms with uniform crossover perform adaptation by exploiting some structure inherent in the fitness distributions arising in practice. Two key questions are i) What is the nature of this structure? and ii) How does a UGA exploit it (i.e. what heuristic does the UGA implicitly implement)? The hyperclimbing hypothesis provides answers to both questions. In doing so it challenges two commonly held views about the conditions necessary for a genetic algorithm to be effective: 1) that the fitness distribution must have a building block structure(Goldberg, 2002; Watson, 2006), 2) that the genetic algorithm must make use of a "linkage learning" mechanism (Goldberg, 2002). Support for the hyperclimbing hypothesis was presented in the proof of concept and validation sections of this article. The computational efficiencies of the UGA identified in an earlier work (Burjorjee, 2009, Chapter 3), and the utility of clamping reported by Huifang and Mo (Huifang and Mo, 2010) lend additional support.

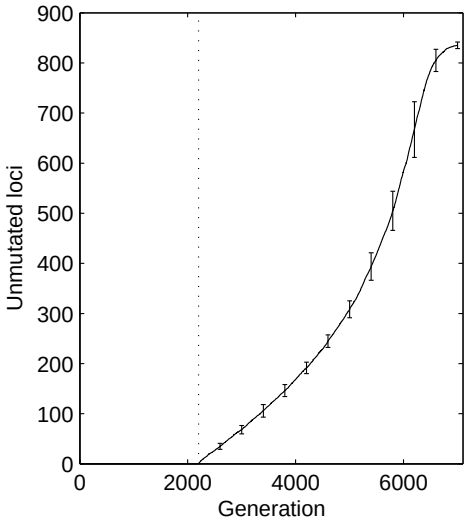
While valuable as an explanation for adaptation in UGAs, the true value of the hyperclimbing hypothesis lies in its generalizability. In a previous work we used the notion of a unit of inheritance—i.e. a gene— to generalize this hypothesis to account for adaptation in simple genetic algorithms with strong linkage between chromosomal loci (Burjorjee, 2009). We conjecture that the hyperclimbing hypothesis can be generalized further to account for adaptation in other kinds of evolutionary algorithms, and that in general, such algorithms perform adaptation by efficiently identifying and progressively fixing globally superior "aspects"—*units of selection* in evolutionary biology speak—of the chromosomes under evolution; in other words, by efficiently decimating the search space. The precise nature of the unit of selection in each case remains to be determined.

Interestingly, the hyperclimbing hypothesis makes a strong contribution to a longstanding debate about the units of selection in *biological populations* (Okasha, 2006; Dawkins, 1999a,b). The material presented in the proof of concept section, and, especially the material in Chapter 3 of an earlier work (Burjorjee, 2009) suggest that the most basic unit of selection is, not the individual gene, but a small set of genes—what we have called a *genoclutch* (Burjorjee, 2009, Section 5.1). Chapter 3 of the earlier work demonstrates conclusively that as a unit of selection, the latter is *not* always reducible to instances of the




(a) Performance of the UGA $Q_c$

(b) Performance of the UGA $Q$




(c) Unmutated Loci in UGA $Q_c$

Figure 5: *(Top:)* The performance, over 10 trials, of the UGA $Q$ (left) and the UGA $Q_c$ (right), on a randomly generated instances 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. *(Bottom:)* 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.

former. In other words, it gives the lie to the common refrain that multi-gene interactions can be ignored when studying adaptation in biological populations because "additive effects are the basis for selection" (Wagner, 2002).

If the hyperclimbing hypothesis and its generalizations are sound we will finally have a *unified* explanation for adaptation in evolutionary algorithms. Rapid advances in the invention, application, and further analysis of these algorithms can be expected to follow. The field of global optimization would be an immediate beneficiary. In turn, a diverse set of fields, including machine learning, drug discovery, and operations research stand to benefit. Take machine learning for instance. Machine learning problems that can be tackled today are, in large part, ones that are reducible *in practice* to convex optimization problems (Bennett and Parrado-Hernández, 2006). The identification of an efficiently implementable, general-purpose meta-heuristic for optimization over rugged, dynamic, and stochastic cost functions—*evolutionary decimation*—promises to significantly extend the reach of the field.

Online Appendix may be downloaded from http://bit.ly/gcABlm

## References


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

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.

Kristin P. Bennett and Emilio Parrado-Hernández. The interplay of optimization and machine learning research. *Journal of Machine Learning Research*, 7:1265–1281, 2006.

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

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

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

Richard Dawkins. *The Extended Phenotype*. Oxford University Press, 1999a.

Richard Dawkins. *The Selfish Gene*. Oxford University Press, 1999b.

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.

D.B. Fogel. *Evolutionary computation: toward a new philosophy of machine intelligence*. Wiley-IEEE Press, 2006. ISBN 0471669512.

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

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

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

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

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 567–570, 2010.

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.

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. URL http://dx.doi.org/10.1007/11919629_62.

Gunar E. Liepins and Michael D. Vose. Characterizing crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 5(1):27–34, 1992.

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

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

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

S. Okasha. *Evolution and the Levels of Selection*. Oxford University Press, USA, 2006.

M. Pelikan, K.G. Helmut, and S. Kobe. Finding ground states of sherrington-kirkpatrick spin glasses with hierarchical boa and genetic algorithms. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 447–454. ACM, 2008.

EM Rudnick, JG Holm, DG Saab, and JH Patel. Application of simple genetic algorithms to sequential circuit test generation. *Proceedings of the European Design and Test Conference*, pages 40–45, 1994.

B. Selman, H.A. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. Citeseer, 1996.

Chris Stephens and Henri Waelbroeck. Schemata evolution and building blocks. *Evolutionary Computation*, 7(2):109–124, 1999.

G. Syswerda. Uniform crossover in genetic algorithms. In J. D. Schaffer, editor, *Proceeding of the Third International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, 1989.

G. Wagner. To epistasis—and beyond! *Evolution*, 56(4):852–855, 2002.

Richard A. Watson. *Compositional Evolution: The Impact of Sex, Symbiosis and Modularity on the Gradualist Framework of Evolution*. The MIT Press, 2006.

Alden H. Wright, Michael D. Vose, and Jonathan E. Rowe. Implicit parallelism. In *GECCO*, 2003.

# Appendices

## Appendix A. Materials and Methods

The pseudocode for the UGA used is given in Algorithm 3. The free parameters of this 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, $\ell$. 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 in this fashion has been shown to reduce sampling error (Baker, 1985; Mitchell, 1996).

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 3 combats this effect by using sigma scaling (Mitchell, 1996, 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 entails that negative fitness values are handled appropriately.

## Appendix B. Analysis of Staircase Functions

Let $\ell$ 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 $\gamma$, 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* $\gamma_1$ *given* $\gamma_2$, denoted $S(\gamma_1 \,|\, \gamma_2)$, to be the difference between the fitness signal of $\gamma_1\gamma_2$ and the fitness signal of $\gamma_2$, 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

**Algorithm 3:** Pseudocode of the SGA used. The population size is an even number denoted by $N$, the length of the chromosomes is $\ell$, 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 $\ell$ 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-MUTATION-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)
    adjustedFitnessValues ← SIGMA-SCALE(fitnessValues)
    parents ← SUS-SELECTION(pop, adjustedFitnessValues)
    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-MUTATION-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
```

$1, \ldots, i-1$). 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 $\lfloor f \rfloor_{j+1}, \ldots, \lfloor f \rfloor_{i-1}$.

For any $i \in [h]$, by Theorem 1 (see below), 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 $\delta$, 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 indirectly 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 above expression follows from the inductive hypothesis, and the second term follows from the definition of a staircase function. Manipulation of this expression yields

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

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. $\square$

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

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 $i = k+1$.

$$\begin{aligned}
&\sum_{\xi \in \Lambda_{k+1} \setminus \{\lceil f \rceil_{k+1}\}} S(\xi) \\
&\quad = \sum_{\psi \in \Gamma_{k+1} \setminus \{\lfloor f \rfloor_{k+1}\}} S(\lceil f \rceil_k \psi) + \sum_{\xi \in \Lambda_k \setminus \{\lceil f \rceil_k\}} \sum_{\psi \in \Gamma_{k+1}} S(\xi\psi) \\
&\quad = \sum_{\psi \in \Gamma_{k+1} \setminus \{\lceil f \rceil_{k+1}\}} S(\lceil f \rceil_k \psi) + \sum_{\psi \in \Gamma_{k+1}} \sum_{\xi \in \Lambda_k \setminus \{\lceil f \rceil_k\}} S(\xi\psi) \\
&\quad = (2^o - 1) S(\lceil f \rceil_k) + 2^o \left( \sum_{\xi \in \Lambda_k \setminus \{\lceil f \rceil_k\}} S(\xi) \right)
\end{aligned}$$

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

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

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 $i \in \{2, \ldots, h\}$,

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

$$= \frac{1}{(2^o)^{i-1}} \left( S(\lceil f \rceil_i) + \sum_{\xi \in \Lambda_{i-1} \setminus \{\lceil f \rceil_{i-1}\}} S(\xi) \right)$$

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

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

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