On the Workings of Genetic Algorithms
The Genoclique Fixing Hypothesis

Keki M. Burjorjee
Computer Science Department
Brandeis University
Waltham, MA 02454
kekib@cs.brandeis.edu

May 18, 2009

Abstract—**We recently reported that the simple genetic algorithm (SGA) is capable of performing a remarkable form of sublinear computation which has a straightforward connection with the general problem of interacting attributes in data-mining. In this paper we explain how the SGA can leverage this computational proficiency to perform efficient adaptation on a broad class of fitness functions. Based on the relative ease with which a practical fitness function might belong to this broad class, we submit a new hypothesis about the workings of genetic algorithms. We explain why our hypothesis is superior to the building block hypothesis, and, by way of empirical validation, we present the results of an experiment in which the use of a simple mechanism called clamping dramatically improved the performance of an SGA with uniform crossover on large, randomly generated instances of the MAX 3-SAT problem.**

I. INTRODUCTION

Genetic algorithms are search heuristics that mimic natural evolution. They have been applied to a wide range of combinatorial optimization problems that are poorly understood, or known to be NP-Hard. While solutions generated by genetic algorithms are often inferior to those yielded by problem-specific search algorithms, in *most* cases specialized search algorithms are not available. When used in such situations, genetic algorithms routinely generate usable solutions relatively quickly.

Unfortunately, the workings of genetic algorithms (GAs) are not well understood. There are several anomalies in the empirical literature that cannot be explained by the *building block hypothesis* [7], [9], [15]—the most comprehensive explanation for the adaptive capacity of genetic algorithms to be proffered to date. Of these anomalies, the two most serious are (i) the widely reported efficacy of uniform crossover [21], [19], [17], and (ii) the unexpected behavior of GAs on Royal Road

functions [16], [6]. In response to such anomalies, and to problems with the theoretical support for the building block hypothesis [5], [18], the building block hypothesis is today treated with a certain amount of skepticism by many GA theorists.

In distancing themselves from the building block hypothesis, several GA theorists have also moved away from the search for a single *comprehensive* explanation for the adaptive capacity of genetic algorithms on practical problems, and have adopted what we shall call a *many little theories* (MLT) approach. This approach is based on the belief that a single theory about the practical workings of genetic algorithms is infeasible because genetic algorithms work in fundamentally different ways depending on, amongst other things, the operators they use, and the classes of practical optimization problems they are applied to. The goal of the MLT approach is to match classes of practical optimization problems with appropriate classes of genetic algorithms. By finding such matches, proponents of this approach hope, eventually, to supply GA practitioners with the means to determine the "right" genetic algorithm for any practical problem.

It seems unlikely that this vision will be realized anytime soon. For a small number of narrowly defined classes of fitness functions, researchers have had some success in deriving upper bounds on the expected number of fitness queries needed to find a global optimum (e.g. [11]). We are unaware, however, of any success in turning such theorems into *theories*, even little ones, with demonstrable practical applications. Another dissatisfying feature of this approach is it's failure, to date, to identify a *computational efficiency* of the genetic algorithm. i.e. a computation of some sort that the genetic algorithm can perform efficiently relative to other known algorithms. Most dissatisfying perhaps, especially to GA practitioners and would-be inventors of more

powerful genetic algorithms, is the basic idea that a single comprehensive account of the practical workings of genetic algorithms is infeasible. Whether one accepts this idea or rejects it is a matter of one's metaphysics; we currently know of no definitive reason for deciding one way or the other. We should mention, however, that a viable comprehensive theory, if one can be found, is preferable, and that historically, scientists have been quite successful at finding viable comprehensive theories for large, internally diverse classes of systems. Most of those who reject the MLT idea continue to subscribe to some version of the building block hypothesis— weak theoretical foundation, and outstanding anomalies notwithstanding. The absence of a promising, *comprehensive* alternative explains the entrenchment of this hypothesis. Presenting such an alternative is the aim of this paper.

In a recent work [3] we reported that the simple genetic algorithm (SGA) possesses a remarkable computational proficiency—a capacity for sublinear computation which, though irrelevant to the problem of global optimization, has straightforward connections with a currently intractable data-mining problem in computational genetics. In this paper, we demonstrate that by applying this computational proficiency recursively, an SGA can perform efficient adaptation on a specific class of fitness functions[1]. Based on this result we infer that by recursively applying this computational proficiency, SGAs can perform efficient adaptation on a very broad class of fitness functions. Given the relative ease with which a practical fitness function might belong to this class of functions, we submit the genoclique fixing hypothesis— a new, comprehensive hypothesis about the practical workings of the simple genetic algorithm—and explain why, as comprehensive hypotheses go, this hypothesis is more promising than the building block hypothesis.

If the genoclique fixing hypothesis is sound, it promises to precipitate significant improvements in the genetic algorithm's capacity for black-box combinatorial optimization. By way of empirical support for this hypothesis we describe what we consider to be the first of such improvements—a mechanism called *clamping*— and present the results of an experiment in which the use of this simple mechanism dramatically improved the performance of a simple genetic algorithm with uniform

---

[1]We believe that the MLT community's inability to identify a computational efficiency of the SGA is a consequence of it's strong focus on global optimization. This focus seems misplaced given that genetic algorithms are valued by practitioners, not for their capacity for efficient global optimization, but for their capacity for efficient *adaptation*.

crossover on large, randomly generated instances of the MAX 3-SAT problem [10].

A. Terminology

We use the word 'gene' to refer to a genomic extent that tends not to be broken up by crossover. This usage accords with Johansen's original use of this word, in 1909, to refer to a "unit of inheritance" [12] [14, p736]. By this definition, a gene is not a strictly defined entity, but has a fading-out quality that is dependent on the expected number of crossover points, and the way these points tend to be distributed over a genome. There is no equivalent concept within genetic algorithmics. The notion of a building block [7] comes close, but since building blocks must, by definition, have above average fitness, whereas a gene need not, the two are not equivalent. It is important to stress that our use of the word gene differs from the way this word typical gets used in genetic algorithmics. Genetic algorithmicists tend to think of two adjacent genomic bits as two separate genes regardless of the crossover operator being used [15], [7]. We regard such bits as separate genes only when crossover is uniform, or close to uniform, i.e. when the expected number of crossover points is approximately half the value of the length of a genome. When the expected number of crossover points is significantly lower, these bits will tend to be inherited together. In this case we regard the two bits as two adjacent "nucleotides" of a single gene.

To ensure a clear comparison between our hypothesis and the building block hypothesis, we now express the latter using the terminology we have just adopted: The building block hypothesis assumes the existence in the initial population of large numbers of genes with statistically significant fitness advantages. According to this hypothesis, adaptation in genetic algorithms is driven by the propagation of such genes, and by the frequent composition in offspring of co-adapted sets of individually advantageous genes that are not co-present in either parent. To avoid confusion, it is important to clarify that by 'co-adapted' we mean something other than the existence of super-additive, or super-multiplicative fitness interactions between the genes concerned; rather, we mean simply that the expected fitness of a genome carrying all the genes in the 'co-adapted' set is greater than the expected fitness of a genome carrying any individual gene in the set; the whole, in other words, is greater than any of the parts.

B. The Basic Idea

We have previously reported [3] that an SGA is capable of efficiently driving a set of co-adapted, *unlinked* genes to fixation even though the fitness signal of this set of genes may be weak relative to the background noise. In driving such genes to fixation the SGA raised the average fitness of the population by a small amount. When any set of genes gets fixed in the population, the representation of the problem space can be thought to have changed. Crucially, the new representation may contain one or more sets of co-adapted genes which may not have had a detectable fitness signal in the old representation. By subsequently driving one or more of these sets to fixation, the SGA can once again "change" it's representation, and in doing so can create new small sets of coadapted genes. And so on.

Each time a small set of co-adapted genes gets fixed, the average fitness of the population will increase by an amount that may be tiny. As the fixation of small sets of co-adapted genes continues, however, these amounts will begin to add up. Based on this thought experiment, we hypothesize that adaptation in genetic algorithms is driven by the iterated "creative fixation" of small sets of co-adapted genes.

II. THE GENOCLIQUE FIXING HYPOTHESIS

Our hypothesis pertains to the class of recombinative SGAs. Our model for this class is the simple genetic algorithm with uniform crossover (UGA). We adopt this algorithm as our model for two reasons: Firstly, under uniform crossover the notion of a unit of inheritance, i.e. a gene, is crisply defined—a gene corresponds exactly with a single bit in a bitstring. This conceptual crispness greatly simplifies our exposition. Secondly, by using suitably crafted classes of fitness functions, the absence of positional bias [4] in uniform crossover can be exploited to demonstrate the computational efficiencies that form the basis for our hypothesis.

A. Mathematical Preliminaries

For any positive integer ℓ, we denote the set of all bitstrings of length ℓ by $\mathfrak{B}_\ell$. We denote a *schema partition* [15] by a tuple consisting of the indices of the defining positions of that schema partition—e.g. $(2, 15, 3)$. The *order* of a schema partition Γ, denoted by $o(\Gamma)$, is the number of elements in some tuple that denotes Γ. Note that a tuple that denotes some schema partition does not have to be ordered; therefore, schema partitions with order greater than one can be denoted in more than one ways. Let Γ_1 and Γ_2 denote two schema partitions. We say that these schema partitions are *orthogonal* if the tuples Γ_1 and Γ_2 have no elements in common. For any genome g, let g_i denote the ith bit of g. For any positive integer n, let $[n]$ denote the set $\{1, \ldots, n\}$. For any genome g of length ℓ and any k-tuple x of distinct integers in $[\ell]$, let $\Xi_x(g)$ denote the bitstring $g_{x_1} \ldots g_{x_k}$. The denotation of a *schema* is dependent on the denotation of the schema-partition that the schema belongs to. Given a schema partition denoted by some tuple Γ, the schemata in this partition are denoted by bitstrings of length $o(\Gamma)$. For any bits $b_1, \ldots, b_{o(\Gamma)}$, the bitstring $b_1 \ldots b_{o(\Gamma)}$ denotes the schema consisting of the genomes $\{g | \Xi_\Gamma(g) = b_1 \ldots b_{o(\Gamma)}\}$. The denotation of the relevant schema partition must always be borne in mind when interpreting a denoted schema.

Let $\Gamma_1 = (x_1, \ldots, x_m)$ and $\Gamma_2 = (y_1, \ldots, y_n)$ denote two orthogonal schema partitions, and let $\gamma_1 = a_1 \ldots a_n$ and $\gamma_2 = b_1 \ldots b_n$ denote schemata of Γ_1 and Γ_2 respectively. Then the concatenation $\Gamma_1\Gamma_2$ denotes the schema partition $(x_1, \ldots x_m, y_1, \ldots, y_n)$, and the concatenation $\gamma_1\gamma_2$ denotes the schema $a_1 \ldots a_m b_1 \ldots b_n$ of $\Gamma_1\Gamma_2$. We will treat the denotation of a schema partition as a tuple sometimes, and as the represented schema partition at others. Likewise, we will treat the denotation of a schema as a bitstring sometimes, and as the represented schema at others. The sense in which we use the denotations of schemata and schema partitions will be clear from the context. For any $m \times n$ matrix M, and for any $i \in [m]$, let $M_{i:}$ denote the n-tuple that the row i^{th} row of M.

B. Staircase Functions

We begin by introducing a class of fitness functions such that given some population of genomes, the co-adaptedness small sets of bits is highly contingent upon the fixation of other bits in the population.

Definition 1. *A staircase function descriptor is a 7-tuple* $(h, o, \delta, \sigma, \ell, L, V)$ *where* h, o *and* ℓ *are positive integers with* $ho \geq \ell$, δ *and* σ *are positive real numbers, and* L *and* V *are matrices with* h *rows and* o *columns such that the values of* V *are binary digits, the elements of* L *are distinct integers from the set* $[\ell]$, *and the rows of* L *are sorted in ascending order.*

Let $\mathcal{N}(a, b)$ denote the normal distribution with mean a and variance b. Then the function described by a staircase function descriptor $(h, o, \delta, \sigma, \ell, L, V)$ is the stochastic function over the set of bitstrings of length ℓ given by algorithm 1. We call h, o, δ, σ, and ℓ the *height,*

Algorithm 1:

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

Input: g is a genome of length ℓ

$y =$ some value drawn from the distribution $\mathcal{N}(0, \sigma^2)$

for *i=1 to h* **do**

    if $\Xi_{L_{i:}}(g) = V_{i1} \ldots V_{io}$ **then**

        $y = y + \delta$

    else

        $y = y - (\delta/(2^o - 1))$

        break

    end

end

return y

---

order, increment, noisiness and *span*, respectively, of the staircase function.

For any $i \in [h]$ we call the schema denoted by $V_{i1} \ldots V_{io}$ of the schema partition denoted by $(L_{i1}, \ldots, L_{io})$ the ith *stage* of the staircase function f. Given the matrix L of the staircase function descriptor, the schema partition of each stage has a canonical denotation. When the staircase function descriptor is clear we will, in the interest of concision, assume that the schema partition of each stage is denoted canonically. Let γ_i denote the ith stage of f. We call the schema denoted by $\gamma_1 \ldots \gamma_i$ the ith *step* of f.

The steps of a staircase function are essentially a progression of nested *hyperplanes* [7, p 53], 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.

Definition 2. *A fractal 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* $[2mn]$, *i.e. each element in* $[2mn]$ *occurs either in* X *or in* Y *once and only once.*

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

Example: Let $(h = 4, o = 2, \delta = 3, \sigma = 1, \ell = 16, L, V)$ be the descriptor of a basic pivotal 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 fractal addressing system such that $X_{1:} = L_{1:}$, $Y_{1:} = L_{2:}$, $X_{2:} = L_{3:}$, and $Y_{2:} = L_{4:}$. A *fractal plot* 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 genome as a greyscale pixel at the genome's fractal address (under the addressing system A). The fitness values returned by f have been scaled linearly to span the range of possible greyscale shades. Lighter shades signify greater fitness. The four steps of f can easily be discerned.

Let us perform a thought experiment in which we generate another fractal 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 be different from that of its homolog in the plot in Figure 1a. Nevertheless, our ability to discern the steps 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; it is easily seen that these values are immaterial to the discernment of the staircase structure of f.

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 fractal plot of f that was obtained using a fractal addressing system $A' = (m = 4, n = 2, X', Y')$ such that $X'_{4:} = L_{1:}$, $Y'_{4:} = L_{2:}$,

---

Algorithm 2: The algorithm for determining the (x, y)-address of a genome under the fractal addressing system (m, n, X, Y). The function $bin2Int$ returns the integer value of a binary string.

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

$granularity = 2^{mn}/2^n$

$x = 1$

$y = 1$

for $i = 1$ *to* m **do**

    $x = x + granularity * bin2Int(\Xi_{X_{i:}}(g))$

    $y = y + granularity * bin2Int(\Xi_{Y_{i:}}(g))$

    $granularity = granularity/2^n$
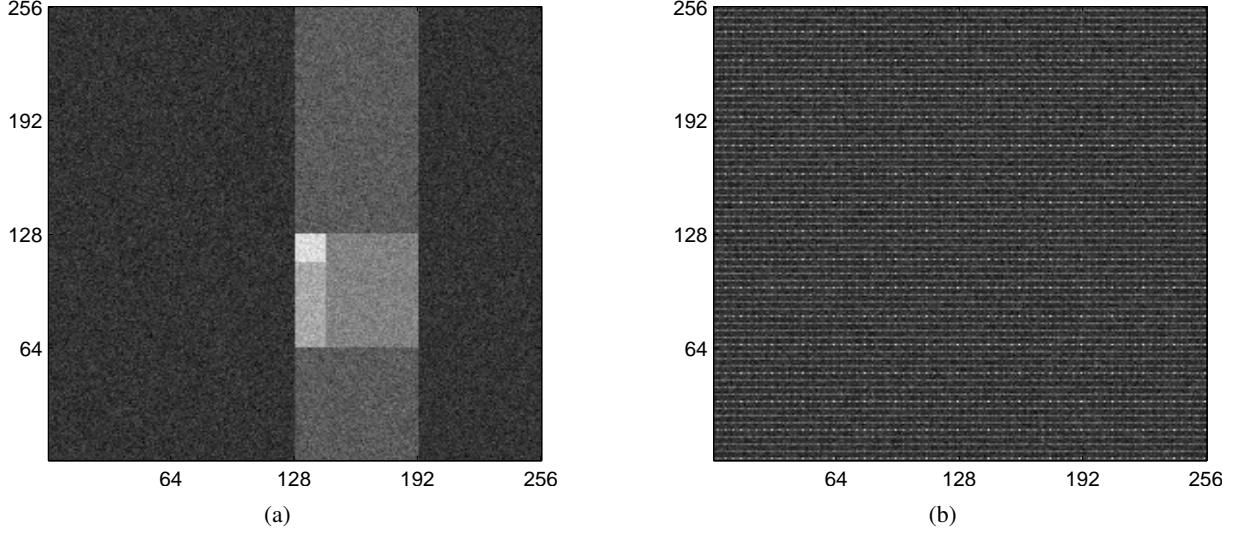
end

return x, y

Fig. 1. A fractal plot of the staircase function f under the fractal addressing systems A (*left*) and A' (*right*)

$X'_{3:} = L_{3:}$, 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 increment to noisiness ratio. 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. As mentioned before, the fitness values returned by the staircase functions are scaled so that they span the range of possible greyscale shades; therefore, had we kept the increment constant and increased the noisiness parameter instead, we would have obtained the same general result as that shown in Figure 2. In general, a decrease in the increment to noisiness ratio of a staircase function results in a decrease in the 'contrast' between the steps of that function.

Let γ denote some schema of the schema partition denoted by Γ. Given some (possibly stochastic) fitness function over a genome set, we define the *fitness signal* of γ, denoted $S_\Gamma(\gamma)$ to be the expected fitness of a genome drawn from the uniform distribution over γ. Let γ_1 and γ_2 be schemata of two orthogonal schema

partitions Γ_1 and Γ_2. We define the *conditional fitness signal of γ_1 given γ_2*, denoted $S_{\Gamma_1|\Gamma_2}(\gamma_1 \,|\, \gamma_2)$, to be the difference between the fitness signal of $\gamma_1\gamma_2$ and the fitness signal of γ_2, i.e. $S_{\Gamma_1|\Gamma_2}(\gamma_1 \,|\, \gamma_2) = S_{\Gamma_1\Gamma_2}(\gamma_1\gamma_2) - S_{\Gamma_2}(\gamma_2)$.

Given a staircase function f with descriptor $(h, o, \delta, \sigma, \ell, L, V)$, we define the *signal to noise ratio* of some schema γ of a schema partition Γ to be $S_\Gamma(\gamma)/\sigma$. Likewise, for any two schemata γ_1 and γ_2 of two orthogonal schema partitions Γ_1 and Γ_2, we define the *conditional signal to noise ratio* of γ_1 given γ_2 to be $S_{\Gamma_1|\Gamma_2}(\gamma_1 \,|\, \gamma_2)/\sigma$.

For any $i \in [h]$, by Lemma 1 (see appendix), the signal to noise ratio of step i is $i\delta/\sigma$. For any $i \in \{2, \ldots, h\}$, corollary 1 of Lemma 1 states that the *conditional* signal to noise ratio of stage i *given* step $(i-1)$ is δ/σ, (a *constant* with respect to i). Finally, for any $i \in [h]$, by Theorem 1, the (unconditional) signal to noise ratio of stage i is

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

Clearly, this ratio decreases rapidly as i increases.

Consider an algorithm that, when given only query access to the staircase function f, can robustly detect the fitness signal of the first step of f, and can restrict future sampling to this step. Observe that the *conditional* signal to noise ratio of the second stage *given* the first step is the same as the signal to noise ratio of the first step. Therefore, if the algorithm restricts its fitness queries to genomes belonging to the first step, it should be able to detect the *conditional* fitness signal of the second
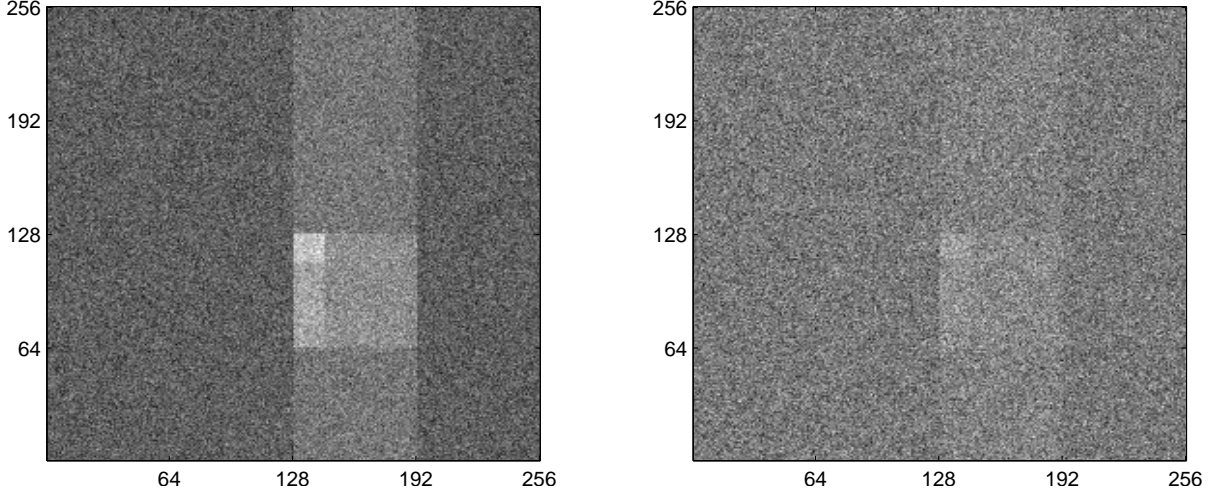
Fig. 2.   Fractal plots under A of two staircase functions, which differ from f only in their increments—1 *(left plot)* and 0.3 *(right plot)* as opposed to 3
.

stage *given* the first step, and should, therefore, be able to identify the second step. Indeed if the algorithm is sufficiently robust it's recursive application need not end with the identification of the second step; higher steps can be identified *indirectly* by identifying lower steps first.

Given expression (1), it is reasonable to suspect that the *direct* identification of step i of a staircase function rapidly becomes computationally infeasible as i increases. The analogy between physical staircases and staircase functions should be transparent; just as it is hard to climb higher steps of a staircase without climbing lower steps first, it becomes computationally infeasible to identify higher steps of a staircase function without identifying lower steps first.

C. Hyperclimbing and Hyperscapes

When an algorithm restricts future queries to some step of a staircase function, we say that it has *climbed* that step. The idea of climbing the steps of a staircase function can be generalized to describe the behavior of arbitrary search algorithms on arbitrary fitness functions (both stochastic and deterministic) over sets of strings. We call the progressive confinement of sampling to hyperplanes of increasing order and increasing expected fitness *hyperclimbing* (short for "hyperplane-climbing"); a search algorithm is said to have *climbed* some hyperplane γ that belongs to some hyperplane partition Γ, if, amongst all the hyperplanes that belong to Γ, future sampling is largely limited to the hyperplane γ.

Hyperclimbing, if it can be implemented efficiently

(a big *if*), seems like a very reasonable way to perform adaptation. Consider some practical fitness function over the set of bitstrings $\mathfrak{B}_\ell$. It is seems reasonable to assume that there exists some low number o_1, such that of the $\binom{\ell}{o_1} \in \Omega(\ell^{o_1})$ ways of partitioning the search space into a set of hyper planes of order o_1, there exists one or more partitions—for the sake of argument let us assume just one—such that this partition contains one or more hyperplanes whose average fitness values are statistically significantly above average under uniform sampling. By restricting future sampling to one of these hyperplanes the hyperclimbing heuristic can increase the expected fitness of all future samples. As far as the hyperclimbing heuristic is concerned, this hyperplane would then comprise the entirety of the search space, i.e. future search can be thought to occur over the space $\mathfrak{B}_{\ell-o_1}$. Our argument now recurses: It seems reasonable to assume that there exists some low number o_2, such that of the $\binom{\ell-o_1}{o_2} \in \Omega((\ell - o_1)^{o_2})$ ways of partitioning the new search space into a set of hyperplanes of order o_2, there exists one or more partitions—for the sake of argument let us assume just one—such that this partition contains one or more hyperplanes whose average fitness values are statistically significantly above average under uniform sampling. By restricting future sampling to one of these hyperplanes, the hyperclimbing heuristic would, once again, increase the expected fitness of all future samples. And so on.

This heuristic will continue to increase the average fitness of the samples it generates as long as there continues to be a way of partitioning the region of the search space that it inhabits into a set of low-order

hyperplanes such that at least one hyperplane in the partition has an average fitness value that is statistically significantly above average under uniform sampling.

Because a hyperclimbing heuristic is sensitive to the "hyperplanar structure" of a search space, not its neighborhood structure, the idea of a landscape [24], [13] is not very helpful when thinking about the behavior of this heuristic. Far more useful is the notion of a *hyperscape*. A hyperscape is like a landscape in that it is just a spatial representation of a fitness function. In a hyperscape, however, the focus is placed, not on the interplay between the fitness function and the neighborhood structure of individual points, but on the statistical fitness properties of individual hyperplanes, and on the spatial relationships between hyperplanes— lower order hyperplanes can *contain* higher order hyperplanes, hyperplanes can *intersect* each other, and disjoint hyperplanes that belong to the same hyperplane partition can be regarded as *parallel*. The use of the concept of a hyperscape in the genetic algorithmics literature can be traced back to the seminal work of Holland [8], who used this concept to reason about the dynamics of recombinative genetic systems. While we disagree with Holland's conclusions, we find hyperscapes to be invaluable in our own reasoning about the dynamics of genetic algorithms—both recombinative and, for reasons that will become clear in section III, non-recombinative.

D. Symmetry Analysis

In a recent work [3] we defined the class of *semi-parameterized* UGAs, and exploited the symmetries of the algorithms in this class to uncover what we consider to be the first two computational efficiencies (albeit highly specific ones) of the SGA to be rigorously identified. The symmetry analysis in that work sets the stage for the symmetry analysis given below. We will show that a semi-parameterized UGA can efficiently climb the first few steps of the staircase functions in a particular class of staircase functions. Remarkably the number of queries required by the semi-parameterized UGA is *independent* of the span of the functions in the class.

Let f be a staircase function with descriptor $(h, o, \delta, \sigma, \ell, L, V)$, 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 four; we therefore write this descriptor as (h, o, δ, σ). Given some staircase function f with descriptor $(h, o, \delta, \sigma, \ell, L, V)$, we define the *basic form*

of f to be the (basic) staircase function with descriptor (h, o, δ, σ).

Let f^* be some basic staircase function with descriptor (h, o, δ, σ), and let F be the set of all staircase functions with basic form f^*. Let W be a semi-parameterized UGA. For any staircase function $f \in F$, let $p_{(f,i)}^{(t)}(x)$ be the probability that the frequency of *stage i* of f in generation t of W^f is x, let $q_{(f,i)}^{(t)}(x)$ be the probability that the frequency of *step i* of f in generation t of W^f is x, and let $r_f^{(t)}$ be the probability that the average fitness of the population of W^f in generation t is x. Then by appreciating the symmetries between the unparameterized UGAs W^{f^*} and W^f we can deduce the following equalities between probability distributions: for any generation t, and for any $i \in [h]$, $p_{(f,i)}^{(t)} = p_{(f^*,i)}^{(t)}$, $q_{(f,i)}^{(t)} = q_{(f^*,i)}^{(t)}$, and $r_f^{(t)} = r_{f^*}^{(t)}$.

Thus, for any generation t, monte-carlo sampling from $r_{f^*}^{(t)}$ is equivalent to monte-carlo sampling from $r_f^{(t)}$, and for any $i \in [h]$, monte-carlo sampling from $p_{(f^*,i)}^{(t)}$, and $q_{(f^*,i)}^{(t)}$ is equivalent to monte-carlo sampling from $p_{(f,i)}^{(t)}$, and $q_{(f,i)}^{(t)}$ respectively.

E. Performance of UGAs on a Staircase Function

Let f_1 be a staircase function with descriptor $(h = 50, o = 4, \delta = 0.3, \sigma = 1)$, and let U denote the semi-parameterized UGA described in the materials and methods section in the appendix. In order to succinctly discuss the results of an experiment in which we applied U to f_1, we introduce the following shorthand: given some population of genomes, the *one-frequency* of some locus is the frequency of the bit 1 at that locus in the population. Figure 3a shows that U is capable of robust adaptation when applied to f_1. Figure 4a shows that under the action of U, the first seven stages of f_1 tend to go to fixation[2] in ascending order. This entails that the first seven steps tend to go to fixation 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. Animation 1, which plots the one-frequencies of all the loci of U^{f_1} in each of 500 generations, shows that the hyperclimbing behavior of U^{f_1} continues beyond the first seven steps. The capacity of U to implement hyperclimbing when applied to f_1 accounts for it's adaptive ability on f_1.

---

[2]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.
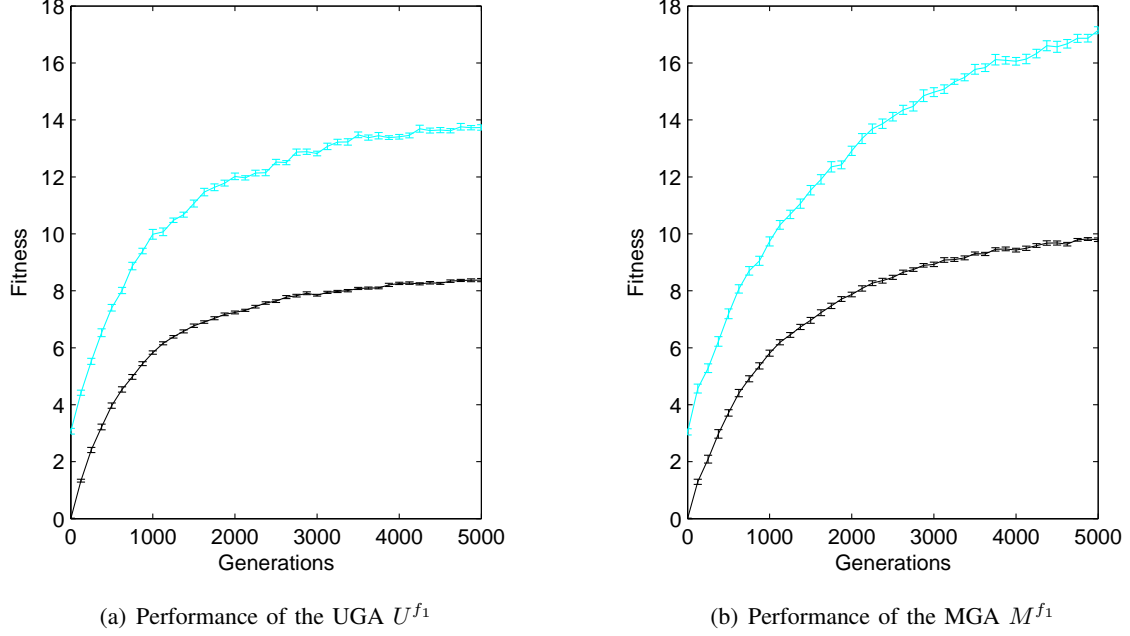
(a) Performance of the UGA U^{f_1}

(b) Performance of the MGA M^{f_1}

Fig. 3. The performance of the semi-parameterized UGA U (left) and the semi-parameterized MGA M (right) on the staircase function f_1 over 20 trials. 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. The error bars show one standard error above and below the mean every 125th generation

Let f be some staircase function with basic form f_1. The conclusions reached in the previous section entail that, had we applied U to f instead of f_1, then *regardless of the span of* f, we would have obtained essentially the same results as those shown in Figures 3a and 4a. This realization is highly remarkable from a computational standpoint.

Consider U's capacity for climbing just the first stage of f. From a computational standpoint, even just this ability is quite remarkable because it is achieved with an expected expenditure of queries that is *constant* in the span of f. We infer that this highly specific capacity for computational efficiency is part of a general capacity of the SGA for efficiently performing what we call *genoclique fixing*. We have previously identified two other highly specific, but nonetheless remarkable, computational efficiencies of the SGA that are instances of its general capacity for efficient genoclique fixing [3]. The results presented here suggest that SGAs can engender robust and efficient adaptation by performing efficient genoclique fixing *recursively*.

F. Mutational Drag and Clamping

Before discussing genoclique fixing, let us contemplate a curious aspect of the behavior of U on f_1. Figure 1 shows that the growth rate of the average fitness of the

population of U^{f_1} decreases as evolution proceeds. To understand this phenomenon consider some genome that belongs to the ith step; the probability that this genome will still belong to step i after mutation is $(1 - x)^{io}$, where x is the per-bit mutation rate. This entails that, U^{f_1} becomes less able to "hold" a population within step i as i increases. In light of this observation, we infer that as i increases the capacity of U^{f_1} to be sensitive to the conditional fitness signal of stage $i + 1$ given step i decreases. This loss in sensitivity explains the decrease in the growth rate of the average fitness of U^{f_1}. We call the "wastage" of fitness queries described here *mutational drag*.

We conceived of the following mechanism for curbing mutational drag in U^{f_1}. This mechanism relies on parameters flagFreq $\in [0, 0.5]$, unflagFreq $\in$ [flagFreq, 0.5], and flagPeriod. If the one-frequency of some locus at the beginning of some generation is less than flagFreq, or greater than $1 -$ flagFreq, then that locus is flagged. Once flagged, a locus remains flagged as long as the one-frequency of the locus is less than unflagFreq, or greater than $1 -$ unflagFreq at the beginning of each subsequent generation. If a flagged locus in some generation t has remained constantly flagged for the last flagPeriod generations, then the locus is considered to have passed our fixation test, and is not mutated in generation t. We
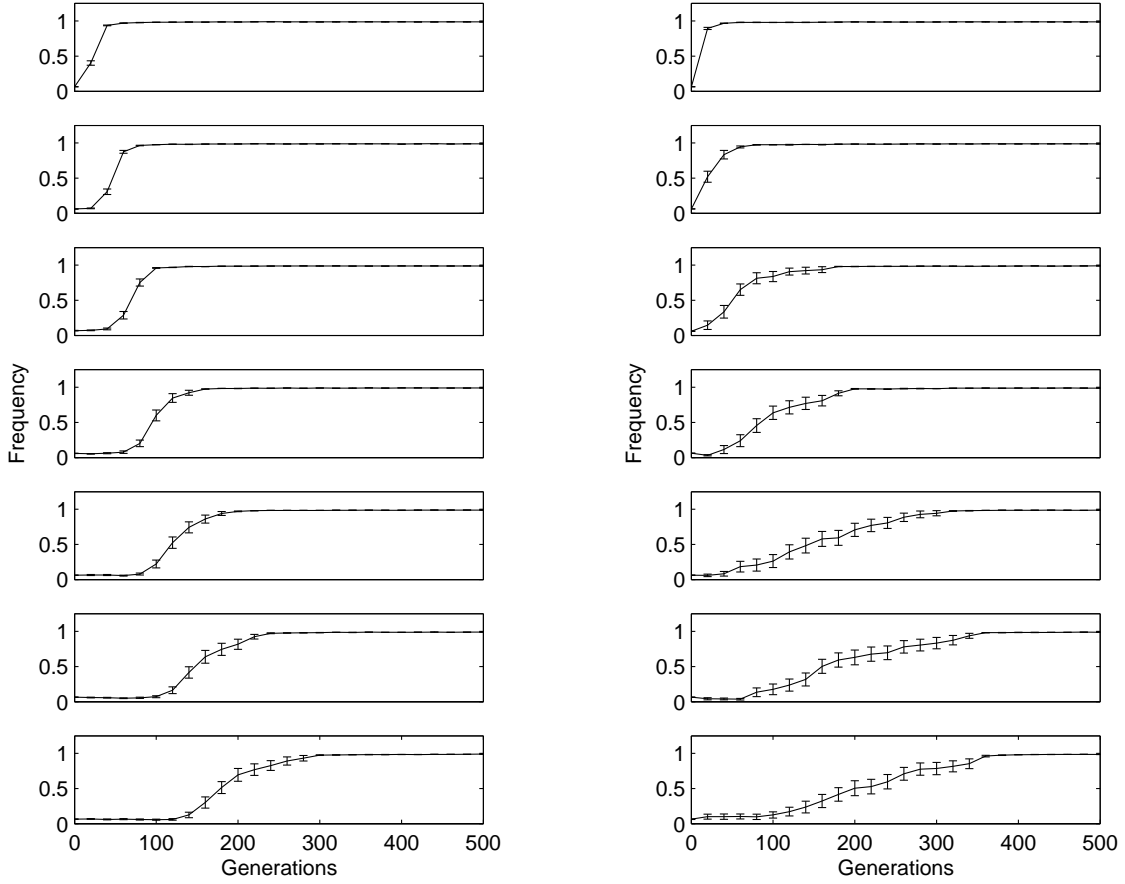
(a) Frequencies of first seven steps of f_1 under the action of U

(b) Frequencies of first seven steps of f_1 under the action of M

Fig. 4. The mean frequency dynamics, over 20 trials, of the first seven steps of the staircase function f_1 (going from the top plot to the bottom plot) under the action of the semi-parameterized UGA U (left), and the semi-parameterized MGA M (right). The error bars show one standard error above and below the mean every twentieth generation

call this mechanism *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 of this locus will get "clamped" at zero or one for the remainder of the run.

We ran a semi-parameterized UGA U_c which used the clamping mechanism described above and was identical to the semi-parameterized UGA U in every other way on the staircase function f_1. The clamping mechanism used by U_c was parameterized as follows: flagFreq $= 0.01$, unflagFreq $= 0.1$, flagPeriod$=200$. The performance of $U_c^{f_1}$ is displayed in figure 5a. Figure 5b 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 steps of f_1. Animation 2 shows the one-frequency dynamics of a single run of $U_c^{f_1}$. The action of the clamping mechanism can be seen in the absence of 'jitter' in the one-frequencies of loci that have been fixed for a while .

G. Genoclique Fixing

We call a small set of co-adaptive genes an *genoclique*. It is important to stress two features of this definition. Firstly, our use of the term "co-adaptive", as opposed to the more conventionally used "co-adapted", is meant to indicate that genocliques are not static entities but dynamic ones that can arise or fade away (become salient, or loose saliency) as the composition of a population of genomes changes. Secondly, note that we have made
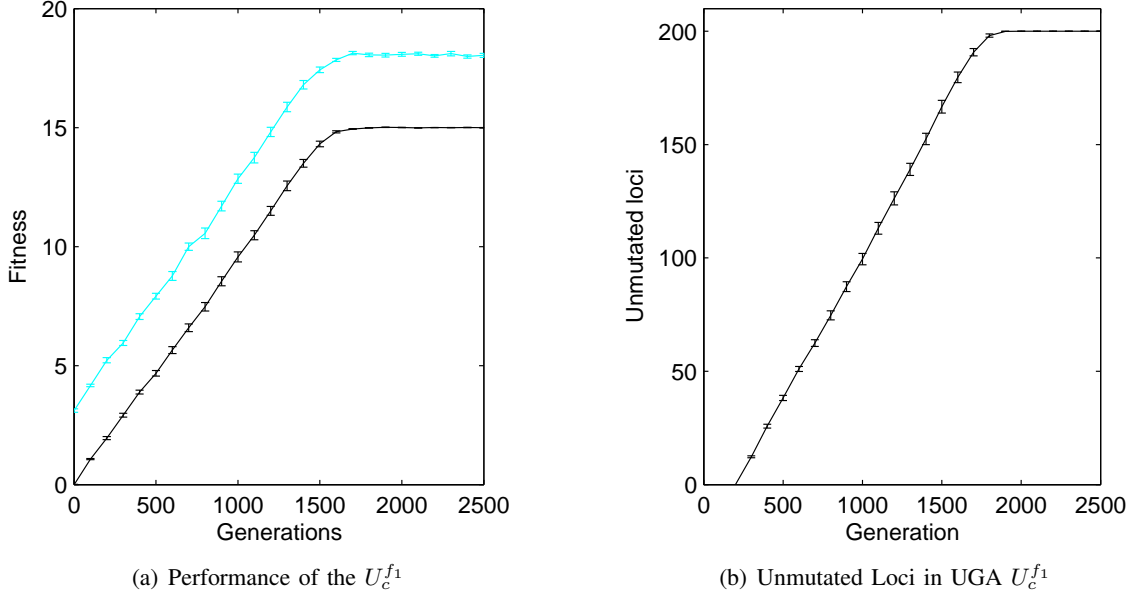
(a) Performance of the $U_c^{f_1}$

(b) Unmutated Loci in UGA $U_c^{f_1}$

Fig. 5. *(Left:)* The performance, over 20 trials, of the semi-parameterized UGA U_c on the staircase function f_1. 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. *(Right:)* The mean number of loci left unmutated by the clamping mechanism. Errorbars show one standard error above and below the mean every hundredth generation

no commitment to the kind of linkage that must exist between the genes in a genoclique. Linkage between such genes can be weak, or even non-existent.

Based on the results in the previous sections, we submit that adaptation in simple recombinative genetic algorithms is driven by the recursive fixing of genocliques. We call this the *genoclique fixing hypothesis*.

This hypothesis rests on assumptions about the distribution of fitness that are easily seen to be *weaker* than those underlying the building block hypothesis [2]—the genoclique fixing hypothesis does not, for example, require large numbers of genes to be individually advantageous at the outset of an evolutionary run. Note, secondly, that genoclique fixing is intuitively a more viable explanation than the building block hypothesis: Because the ability of recombination to disrupt a genoclique declines rapidly as the genoclique goes to fixation, it is easy to see how the fixing of genocliques can be a robust vehicle for adaptation in recombinative genetic systems; in comparison it is much more difficult to grasp how synergistic composition can be a robust vehicle for adaptation. After all, though recombination can occasion the synergistic composition of genes, it can also occasion the destruction of such compositions. Thirdly, note that unlike the building block hypothesis, for which no proof of concept has been provided in over three decades, the genoclique fixing hypothesis is accompanied by proof of concept (see the previous section) from the start.

H. Emprical Validation

We now present the results of an experiment in which the use of clamping dramatically improved the performance of a UGA on large, randomly generated instances of the MAX 3-SAT problem. This difference in performance strongly supports our hypothesis.

We ran two semi-parameterized UGAs—one with clamping (Q_c), and one without (Q)—on randomly generated instances of the MAX 3-SAT problem [10] with 10,000 binary variables and 50,000 clauses. Both UGAs used a straightforward encoding in which each bit of a genome represents the value of a single MAXSAT variable. The fitness of a genome was simply the number of clauses satisfied under the variable assignment represented by the genome. The clamping mechanism used by Q_c was parameterized as follows: `flagFreq = 0.01`, `unflagFreq = 0.1`, `flagPeriod=200`. Figure 6c shows the number of loci that this mechanism left unmutated in each generation. By the four thousandth generation, the clamping mechanism left on average over 2500 loci unmutated. Given any set of 2500 loci, in the absence of clamping the chance that the 2500 loci will all go unmutated in some genome is $0.997^{2500} < 5.5 \times 10^{-4}$. The "drag" resulting from the continued mutation of

long-fixed loci in Q explains why this UGA was outperformed by Q_c (Figure 6a,b). The difference between the mean best-of-population fitness of the final generation of Q_c and the mean best-of-population fitness of the final generation of Q was 1148.5 clauses. By all indications, this difference would have been larger had we allowed our trials to continue past 4000 generations.

III. ON THE FUNCTION OF RECOMBINATION

Under the building block hypothesis, the function of recombination is clear—to drive adaptation by effecting the synergistic composition of advantageous genes, and co-adapted sets of advantageous genes. If genoclique fixing, not synergistic composition, is the vehicle for adaptation, then the function of recombination is less transparent. If the genoclique fixing hypothesis is to be a viable alternative to the building block hypothesis, the advantage that recombination often confers must be accounted for.

Under the genoclique fixing hypothesis, the widely reported efficacy of recombination, especially strong forms of recombination, like uniform crossover, actually seems anomalous. As the expected number of crossover points increases, the size of the genes in a genoclique decreases, and the number of genes in a genoclique therefore tends to decrease. Since genocliques with fewer genes are less likely to be disrupted by recombination, and since the disruption of genocliques *hampers* their fixation, it seems like the fewer the expected number of crossover points in a crossover operation, the better.

The phenomenon of hitchhiking [20], [6] seems to offer an easy explanatory escape from this anomaly. It is simple to see how, as the size of the genes in a genoclique increases, some situated bit b can become part of one or more genocliques even though it does not contribute to the co-adaptivity of any of these genocliques. If any of the genocliques go to fixation then so will b (i.e. b will *hitchhike* to fixation). Now, suppose it so happens that the complement of b is implicated in the co-adaptivity of one or more genocliques later on in the evolutionary run. It seems reasonable to suspect that the prior *spurious fixation* of b will prevent any genocliques containing the complement of b from going to fixation.

Since the prevalence of hitchhiking increases in inverse relation to the expected number of crossover points, it seems plausible that the relative absence of hitchhiking in UGAs can account for the widely reported efficacy of uniform crossover. The prevalence of hitchhiking will be most extreme when recombination is entirely absent. To test our hunch about the utility of recombination, we therefore switched off crossover in the semi-parameterized UGA U and applied the resulting semi-parameterized mutation-only simple genetic algorithm (MGA), denoted M, to the staircase function f_1. A comparison between Animations 1, and 3 confirms the prevalence of hitchhiking in M^{f_1} (note how the one-frequencies of high-index loci rush to one or zero at the beginning of the run even though selection is not acting at these loci), and it's relative absence in U^{f_1} (while the one-frequencies of high-index loci do diverge from 0.5, they do so relatively slowly). Remarkably, despite the prevalence of hitchhiking, M^{f_1} outperforms U^{f_1} (compare Figure 3b with fig 3a). Figure 4b, and Animation 3 show that, like U^{f_1}, M^{f_1} performs adaptation by implementing hyperclimbing. The difference in performance seen when comparing Figure 3a with Figure 3b turns out to be representative of a systematic difference in the performance of UGAs and MGAs on basic staircase functions. In an informal empirical comparison of the performance of these SGAs over a broad parametric regime we found that switching off recombination typically *improves* performance.

The implications of these results for the genoclique fixing hypothesis are mixed. On the one hand, the "easy explanatory escape" that hitchhiking seemed to offer turns out not to be quite so easy. If anything, the widely observed efficacy of recombination is now *more* puzzling than before.

On the other hand, the observed hyperclimbing behavior of MGAs on staircase functions reveals the centrality of *fixing* to adaptation in *all* SGAs. To see why, observe that the conclusions we reached by exploiting the symmetries of unparameterized UGAs with staircase fitness functions hold even when uniform crossover is switched off. This realization entails that MGAs, like UGAs, are capable of efficient hyperclimbing[3]. In terms of the expected number of crossover points per crossover operation, MGAs and UGAs occur at opposite ends of a continuum. Since both these SGAs are capable of efficient hyperclimbing, hyperclimbing seems well positioned to serve as the organizing idea for the study of adaptation in *all* SGAs.

---

[3]The building block hypothesis is decidedly silent when it comes to explaining the adaptive capacity of non-recombinative genetic algorithms [5, p147-155]. With the discovery that MGAs can implement efficient hyperclimbing, these reports can now be accounted for.
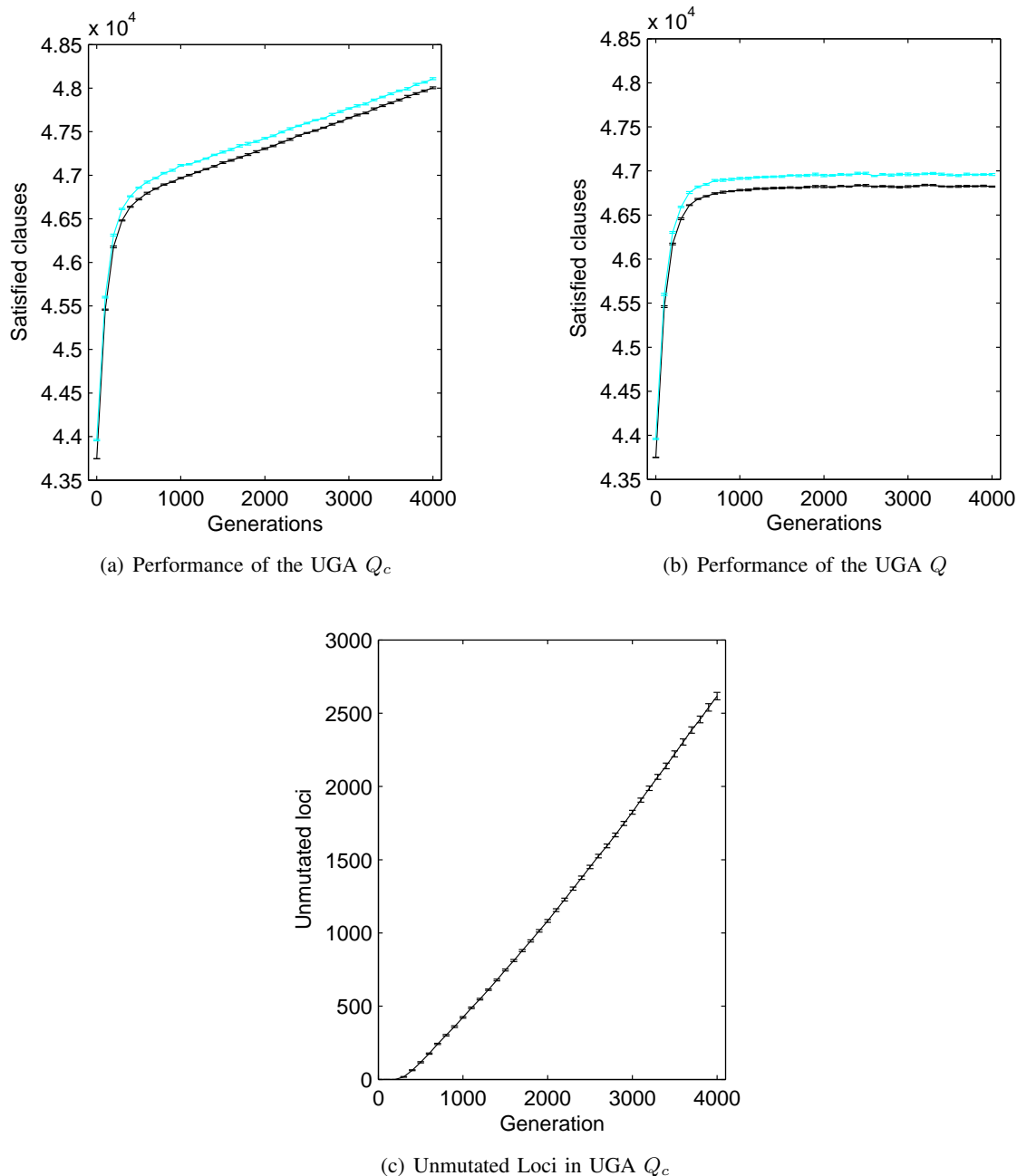
(a) Performance of the UGA Q_c

(b) Performance of the UGA Q

(c) Unmutated Loci in UGA Q_c

Fig. 6. *(Top:)* The performance, over 10 trials, of the UGA Q_c (left) and the UGA Q (right), on randomly generated instances of the MAX 3-SAT problem with 10,000 variables and 50,000 clauses. Q_c used clamping, whereas Q did not. 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. *(Bottom:)* The mean number of loci left unmutated by the clamping mechanism of Q_c. Errorbars show one standard error above and below the mean every hundredth generation

A. Multi-Staircase Functions

Returning to the task of explaining the function of recombination, we conjecture that staircase functions, illuminative as they are, fail to capture some key feature that is commonly present in fitness distributions induced through the representational choices of GA practitioners. We conjecture, furthermore, that hitchhiking interferes with an MGA's ability to exploit this feature.

Observe that when a UGA is applied to a staircase function, genocliques will tend to become salient *sequentially*. This need not be true when recombinative SGAs are applied to real-world problems. Might hitchhiking pose more of a problem when genocliques become salient *concurrently*? To test this hunch we conceived of the class of *multi-staircase functions*—a straightforward generalization of the class of staircase functions.

Algorithm 3: A multi-staircase function with descriptor
$(c, h, o, \delta, \sigma, \ell, L^{(1)}, \ldots, L^{(c)}, V^{(1)}, \ldots, V^{(c)})$

Input: g is a genome of length ℓ
$y =$ some value drawn from the distribution $\mathcal{N}(0, \sigma^2)$
for j=1 to c **do**
    for i=1 to h **do**
        if $\left(g_{L_{i1}^{(j)}} = V_{i1}^{(j)}\right) \wedge \ldots \wedge \left(g_{L_{io}^{(j)}} = V_{io}^{(j)}\right)$
        then
            $y = y + \delta$
        else
            $y = y - (\delta/(2^o - 1))$
            break
        end
    end
end
return y

Definition 3. *A multi-staircase function descriptor is a tuple* $(c, h, o, \delta, \sigma, \ell, L^{(1)}, \ldots, L^{(c)}, V^{(1)}, \ldots, V^{(c)})$ *where* $c, h,$ o *and* ℓ *are positive integers with* $cho \leq \ell$, δ *and* σ *are positive real numbers, and* $L^{(1)}, \ldots, L^{(c)}$ *and* $V^{(1)}, \ldots, V^{(c)}$ *are matrices with* h *rows and* o *columns such that the elements of* $L^{(1)}, \ldots, L^{(c)}$ *are distinct integers from the set* $[\ell]$ *(i.e.* $L_{i_1 j_1}^{(k_1)} \neq L_{i_2 j_2}^{(k_2)}$ *unless* $i_1 = i_2 \wedge j_1 = j_2 \wedge k_1 = k_2$*), each row in each of the matrices* $L^{(1)}, \ldots, L^{(c)}$ *is sorted in ascending order, and the elements of* $V^{(1)}, \ldots, V^{(c)}$ *are binary digits.*

The function described by a multi-staircase function descriptor $(h, o, \delta, \sigma, \ell, L^{(1)}, \ldots, L^{(c)}, V^{(1)}, \ldots, V^{(c)})$ is the stochastic function over the set of bitstrings of length ℓ given by algorithm 1. We call c the *cardinality* of the multi-staircase function, Like we did with staircase functions, we call h, o, δ, σ, and ℓ the *height, order, increment, noisiness* and *span* respectively.

Our analogy between staircases and staircase functions can be extended to apply to multi-staircase functions. When the cardinality of a multi-staircase function is one, a single staircase is induced; when the cardinality is two or more, multiple staircases are induced. In the latter case, loci belonging to the steps of a particular staircase may be scattered amongst loci belonging to the steps of other staircases. However, since each locus belongs to no more than one staircase, and since the fitness benefits of climbing separate staircases combine additively, each staircase may be climbed independently; in other words, the "next step" of several staircases can become salient *concurrently*. The "degree" of concurrency is determined by the cardinality of the multi-staircase function.

B. Symmetry Analysis

Let f be a multi-staircase function with descriptor $(c, h, o, \delta, \sigma, \ell, L^{(1)}, \ldots, L^{(c)}, V^{(1)}, \ldots, V^{(c)})$. We say that this function is *basic* if $\ell = cho$, $L_{ij}^{(k)} = ho(k - 1) + o(i - 1) + j$, i.e. $L^{(k)}$ is the matrix of integers from $(ho)(k - 1) + 1$ to hok laid out row-wise, and V is a matrix of ones. If f is basic, then the first five elements of the descriptor of f determines the remaining elements; we therefore write this descriptor as $(c, h, o, \delta, \sigma)$. Given some multi-staircase function f with descriptor $(c, h, o, \delta, \sigma, \ell, L^{(1)}, \ldots, L^{(c)}, V^{(1)}, \ldots, V^{(c)})$, we define the *basic form* of f to be the basic multi-staircase function $(c, h, o, \delta, \sigma)$.

Let f^* be some basic staircase function with descriptor $(c, h, o, \delta, \sigma)$, and let F be the set of all staircase functions with basic form f^*. Let W be a semi-parameterized UGA or a semi-parameterized MGA. For any staircase function $f \in F$, let $r_f^{(t)}$ be the probability that the average fitness of the population of W^f in generation t is x. Then by appreciating the symmetries between the unparameterized UGAs W^{f^*} and W^f we can deduce the following equalities between probability distributions: for any generation t, $r_f^{(t)} = r_{f^*}^{(t)}$. Thus, for any generation t, monte-carlo sampling from $r_{f^*}^{(t)}$, is equivalent to monte-carlo sampling from $r_f^{(t)}$.

C. Performance of a UGA and an MGA on a Multi-Staircase Function

Let f_2 denote a multi-staircase fitness function with descriptor $(c = 10, h = 50, o = 4, \delta = 0.3, \sigma = 1)$. Figure 7 shows that when applied to this function, on average the semi-parameterized UGA U outperforms the semi-parameterized MGA M. Animations 3 and 4 show the one-frequency dynamics of U^{f_2} and M^{f_2} in a single run of each. These animations qualitatively show that U^{f_2} is better than M^{f_2} at climbing the ten staircases of f_2 *in parallel*. The prevalence of hitchhiking in M^{f_2}, and it's relative absence in U^{f_2} seems, at least qualitatively, to account for this difference in ability.

D. Concurrent Genoclique Fixing

We emphasize that the semi-parameterized SGAs U and M mentioned above are the same semi-parameterized SGAs that were used in our previous experiments. Recall that on average M outperformed U when applied to the basic staircase function f_1. This function can be thought of as a basic *multi*-staircase

(a) Performance of the UGA U^{f_2}

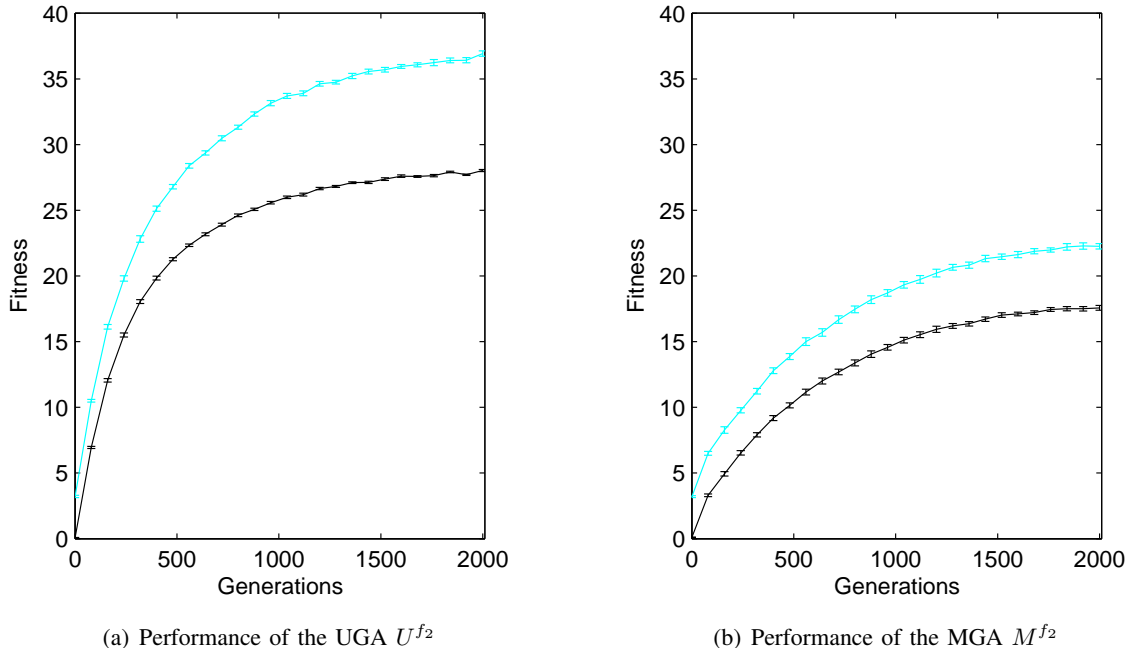(b) Performance of the MGA M^{f_2}

Fig. 7. The performance of the semi-parameterized UGA U (left) and the semi-parameterized MGA M (right) on the multi-staircase function f_2 over 20 trials. 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. The error bars show one standard error above and below the mean every 80th generation

function with cardinality one. When f_1 is regarded as such, the difference between it and f_2 amounts solely to a difference in cardinality. Based on these observations, and the results mentioned above, we submit that the function of recombination in genetic algorithms is to reduce hitchhiking; by reducing hitchhiking, recombination allows the fixing of genocliques to proceed concurrently.

IV. CONCLUSION AND OUTLOOK

Many details of the new theory presented in this paper remain to be worked out and/or expressed. For example, the function of mutation needs to be explained (if mutation causes drag, why use it?), and the relationship between population size and a recombinative SGA's capacity for efficient genoclique fixing merits attention. Presenting a complete account of the workings of recombinative SGAs, however, is not our aim. Rather, we have sought to present a *general* account of these workings, and to support this account in ways that make it compelling, or, to be precise, more compelling than the building block hypothesis—to date, the only other general account of the practical workings of recombinative SGAs.

Perhaps the best way to understand the difference between the building block hypothesis and the genoclique

fixing hypothesis is by focusing on the part played by fixation in each account. In downplaying the role of fixation, the building block hypothesis departs rather radically from the accounts about adaptation in biological populations that one finds in population genetics. The building block hypothesis holds that genetic algorithms work by maintaining a store of partial solutions—advantageous genes, and coadapted sets of individually advantageous genes—and by hierarchically assembling these partial solutions as evolution proceeds. Crucially, the building block hypothesis is not opposed to the idea that an advantageous gene and it's advantageous bitwise complement can both persist in an evolving population. Indeed, as Watson's work with hierarchical if and only if functions [22], [23] indicates, the persistence of such alleles is *expected*. Because the building block hypothesis dispenses with fixation, it needs to look to the weakness of recombination for a vehicle for "locking in" adaptive gains. This hypothesis cannot, therefore, explain the widely observed adaptive capacity of SGAs with strong forms of recombination (e.g. uniform crossover)

In contrast, the genoclique fixing hypothesis holds that *fixation* is the vehicle by which adaptive gains are locked in. The genoclique fixing hypothesis is based on the key realization that selection can drive a small set of unlinked coadapted genes to fixation even as these genes

are repeatedly separated by recombination whenever they co-occur [3]. Once such a set of genes—what we call a *genoclique*—has gone to fixation, recombination looses it's power to disrupt this set, and the fitness advantage that the genoclique confers, even if it is only a small increase in expected fitness, gets locked in. Since recombination is not required to "protect" genocliques as they go to fixation, the genoclique fixing hypothesis has no problem in accounting for the adaptive capacity of UGAs. So, while the building block hypothesis can only account for the adaptive capacity of SGAs with small numbers of crossover points, the genoclique fixing hypothesis can account for the adaptive capacity of *any* recombinative SGA.

The genoclique fixing hypothesis can be thought of as a particular instantiation of a more general *unified* theory about the practical workings of *all* SGAs, including ones that do not use crossover. In section II-C we introduced the idea of a hyperclimbing heuristic. This heuristic is sensitive, not to the local features of a search space, but to fitness properties of the hyperplanes of the space. The hyperclimbing heuristic is therefore not susceptible to the typical problems affecting local search algorithms (e.g. entrapment in the fitness basins of local optima). While hyperclimbing seems like a reasonable way to perform adaptive search, the moment one factors in what appears to be the high cost, in terms of time and fitness queries, of implementing this heuristic, it quickly looses it's shine. Our exciting discovery—the crux of this paper—is that simple genetic algorithms can implement hyperclimbing efficiently.

On the problems studied, we found that an SGA with uniform crossover, and an SGA without crossover can both perform efficient hyperclimbing. Uniform crossover and no crossover are, in terms of expected number of crossover points, at opposite ends of the "crossover continuum" of an SGA. We therefore infer that a capacity for efficient hyperclimbing underlies the adaptive capacity of all SGAs. We believe that this idea—the *hyperclimbing thesis*—can serve as a platform for the unified study of adaptation in all genetic algorithms.

References

[1] 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.

[2] Keki M. Burjorjee. The fundamental problem with the building block hypothesis. *CoRR*, abs/0810.3356, 2008.

[3] Keki M. Burjorjee. Two remarkable computational competencies of the simple genetic algorithm. *CoRR*, abs/0810.3357, 2008.

[4] 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.

[5] David B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, NY, 2000.

[6] Stephanie Forrest and Melanie Mitchell. Relative building-block fitness and the building-block hypothesis. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 109–126, San Mateo, CA, 1993. Morgan Kaufmann.

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

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

[9] John H. Holland. Genetic algorithms. *Scientific American*, 267, 1992.

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

[11] Thomas Jansen and Ingo Wegener. Real royal road functions–where crossover provably is essential. *Discrete Applied Mathematics*, 149(1-3):111–125, 2005.

[12] W. Johansen. *Elemente der Exakten Erblichkeitslehre*. Jena: Gustav Fisher, 1909.

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

[14] Ernst Mayr. *The growth of biological thought*. Belknap Press of Harvard University Press, 2003.

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

[16] Melanie Mitchell, Stephanie Forrest, and John H. Holland. The royal road for genetic algorithms: Fitness landscapes and GA performance. In F. J. Varela and P. Bourgine, editors, *Proc. of the First European Conference on Artificial Life*, pages 245–254, Cambridge, MA, 1992. MIT Press.

[17] 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.

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

[19] 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.

[20] J. David Schaffer, Larry J. Eshelman, and Daniel Offut. Spurious correlations and premature convergence in genetic algorithms. In Gregory J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 102–112, San Mateo, 1991. Morgan Kaufmann.

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

[22] Richard A. Watson. *Compositional Evolution: Interdisciplinary Investigations in Evolvability, Modularity, and Symbiosis*. PhD thesis, April 01 2002.

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

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

APPENDIX

MATERIALS AND METHODS

The semi-parameterized SGA denoted by U was implemented with an SGA that is faithful to the specification for a simple genetic algorithm given by Mitchell [15, p 10] in every way, except for the following two:

1) In each generation, right after evaluating the fitness of all individuals, our SGA used sigma scaling [15, p 167] to adjust the fitness of each individual, and used this adjusted fitness when selecting the parents of that generation. Suppose $f_x^{(t)}$ is the fitness of some individual x in some generation t, and suppose the average fitness and standard deviation of the fitness of the individuals in generation t are given by $\overline{f^{(t)}}$ and $\sigma^{(t)}$ 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,

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

2) The SGA used universal stochastic stochastic sampling [1] [15, p 166] to select parents.

Selection was fitness-proportionate. The population size was 500. Bit-flip mutation with a mutation rate of 0.003 per bit was used. The probability of crossover was one.

The population size of the semi-parameterized UGAs Q_c and Q was 200. Q_c used clamping (described in the main text), whereas Q did not. Other than the population size, and the use of clamping, Q_c and Q were the same in every way to the semi-parameterized UGA U. The SGA used to implement the semi-parameterized SGAs described above was written in Matlab and is available for download[4].

PROOFS

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

PROOF: The proof is by induction on i. The base case, when $i = h$ is easily seen to be true. 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 γ_j denote stage j, and let Γ_j be the canonical denotation of the schema partition containing γ_j. The fitness signal of step $k - 1$ is given by

$$\frac{1}{2^o} \left(S_{\Gamma_1 \ldots \Gamma_k}(\gamma_1 \ldots \gamma_k) + \sum_{\psi \in \Gamma_k \setminus \{\gamma_k\}} S_{\Gamma_1 \ldots \Gamma_k}(\gamma_1 \ldots \gamma_{k-1}\psi) \right)$$
$$= \frac{\delta k}{2^o} + \frac{2^o - 1}{2^o} \left(\delta(k-1) - \frac{\delta}{2^o - 1} \right)$$

where the first term of the right hand side follows from the inductive hypothesis. Manipulation of the right hand side yields

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

which upon further manipulation yields $(k-1)\delta$ $\square$

Corallary 1. *For any $i \in \{2, \ldots, h\}$, the conditional signal to noise ratio of stage i given step $i - 1$ is δ/σ*

PROOF The conditional signal to noise ratio of stage i given step $i - 1$ is given by

$$S_{\Gamma_i | \Gamma_1 \ldots \Gamma_{i-1}}(\gamma_i | \gamma_1 \ldots \gamma_{i-1})/\sigma$$
$$= (S_{\Gamma_1 \ldots \Gamma_i}(\gamma_1 \ldots \gamma_i) - S_{\Gamma_1 \ldots \Gamma_{i-1}}(\gamma_1 \ldots \gamma_{i-1}))/\sigma$$
$$= (i\delta - (i-1)\delta)/\sigma$$
$$= \delta/\sigma \ \square$$

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

PROOF: For any $j \in [h]$, let γ_j denote stage j, and let Γ_j be the canonical denotation of the partition containing γ_j. We first prove the following claim

Claim 1. *For any $i \in [h]$,*

$$\sum_{\xi_1 \in \Gamma_1 \ldots \Gamma_i \setminus \{\gamma_1 \ldots \gamma_i\}} S_{\Gamma_1 \ldots \Gamma_i}(\xi) = -i\delta$$

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

$$\sum_{\xi \in \Gamma_1 \setminus \{\gamma_1\}} S_{\Gamma_1}(\xi) = (2^o - 1)\left(\frac{-\delta}{2^o - 1}\right) = -\delta$$

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

$$\sum_{\xi_1 \in \Gamma_1 \ldots \Gamma_i \setminus \{\gamma_1 \ldots \gamma_{k+1}\}} S_{\Gamma_1 \ldots \Gamma_{k+1}}(\xi)$$

$$= \sum_{\psi \in \Gamma_{k+1} \setminus \{\gamma_{k+1}\}} S_{\Gamma_1 \ldots \Gamma_k +1}(\gamma_1 \ldots \gamma_k \psi) \; + $$

$$\sum_{\xi \in \Gamma_1 \ldots \Gamma_k \setminus \{\gamma_1 \ldots \gamma_k\}} \sum_{\psi \in \Gamma_{k+1}} S_{\Gamma_1 \ldots \Gamma_k +1}(\xi \psi)$$

$$= \sum_{\psi \in \Gamma_{k+1} \setminus \{\gamma_{k+1}\}} S_{\Gamma_1 \ldots \Gamma_k +1}(\gamma_1 \ldots \gamma_k \psi) \; + $$

$$\sum_{\psi \in \Gamma_{k+1}} \sum_{\xi \in \Gamma_1 \ldots \Gamma_k \setminus \{\gamma_1 \ldots \gamma_k\}} S_{\Gamma_1 \ldots \Gamma_k +1}(\xi \psi)$$

$$= (2^o - 1) \, S_{\Gamma_1 \ldots \Gamma_k +1}(\gamma_1 \ldots \gamma_k) \; + $$

$$2^o \left(\sum_{\xi \in \Gamma_1 \ldots \Gamma_k \setminus \{\gamma_1 \ldots \gamma_k\}} S_{\Gamma_1 \ldots \Gamma_k +1}(\xi) \right)$$

where the last equality follows 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

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

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

For a proof of the theorem, observe that stage 1 and step 1 are the same schema. So, by Lemma 1,

$S_{\Gamma_1}(\gamma_1) = \delta$. Thus the theorem holds for $i = 1$. For any $i \in \{2, \ldots, h\}$,

$$S_{\Gamma_k}(\gamma_i) = \frac{1}{(2^o)^{i-1}} \left(S_{\Gamma_1 \ldots \Gamma_i}(\gamma_1 \ldots \gamma_i) \; + \right.$$

$$\left. \sum_{\xi \in \Gamma_1 \ldots \Gamma_{i-1} \setminus \{\gamma_1 \ldots \gamma_{i-1}\}} S_{\Gamma_1 \ldots \Gamma_i}(\xi \gamma_k) \right)$$

$$= \frac{1}{(2^o)^{i-1}} \left(S_{\Gamma_1 \ldots \Gamma_i}(\gamma_1 \ldots \gamma_i) \; + \right.$$

$$\left. \sum_{\xi \in \Gamma_1 \ldots \Gamma_{i-1} \setminus \{\gamma_1 \ldots \gamma_{i-1}\}} S_{\Gamma_1 \ldots \Gamma_{i-1}}(\xi) \right)$$

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

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

$$= \frac{\delta}{(2^o)^{i-1}} \quad \square$$
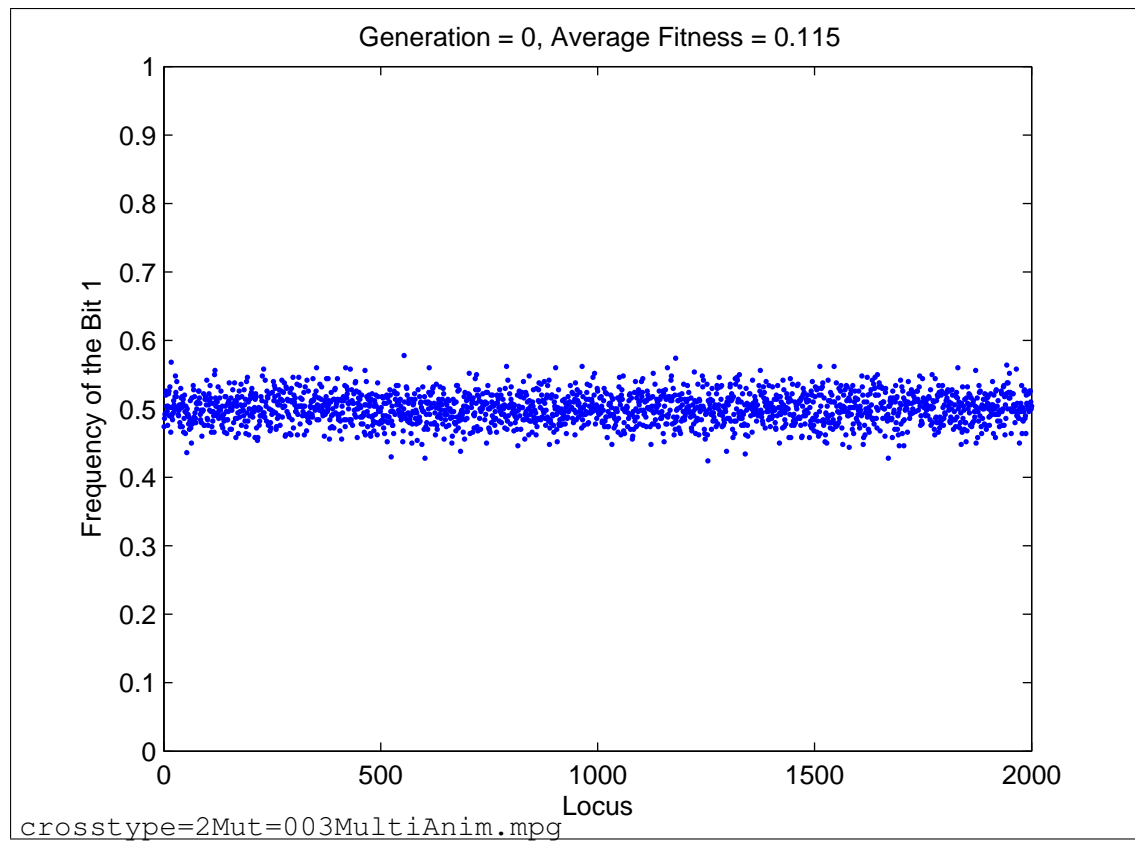
[Page Intentionally Left Blank]
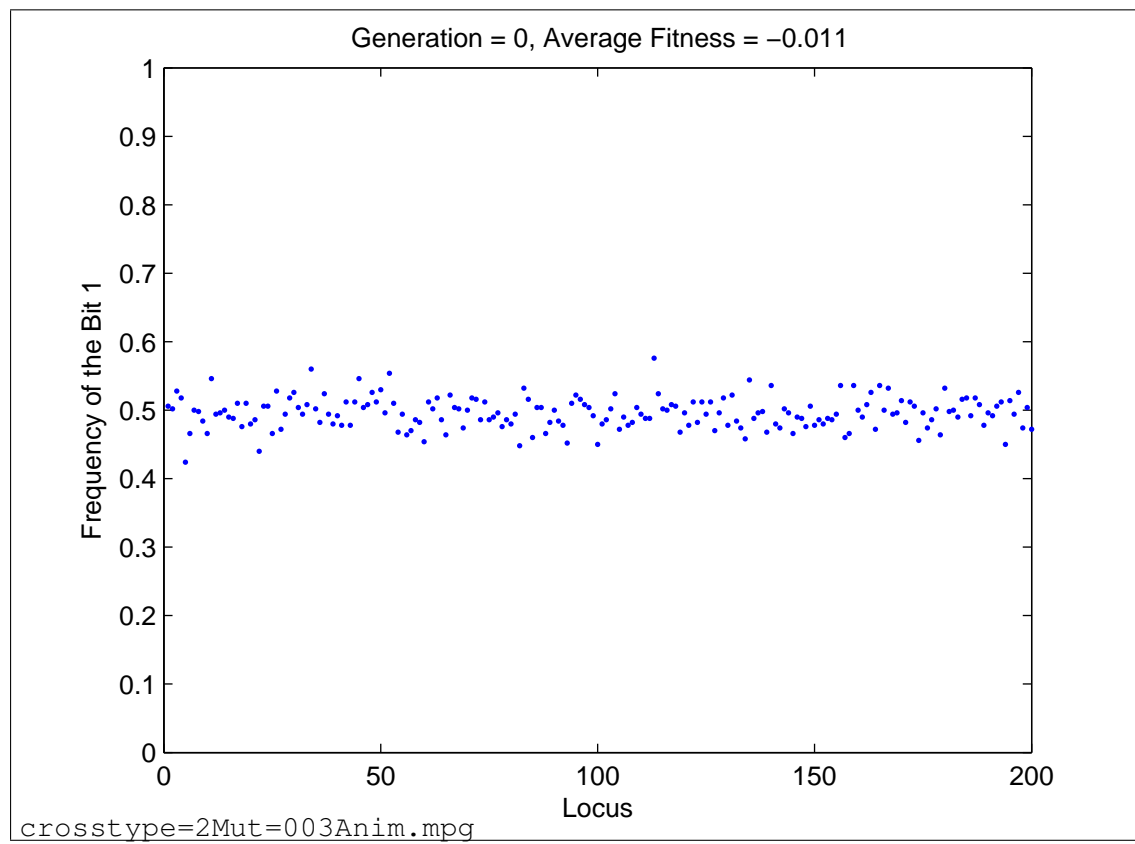
Animation 1: [Click on image to play] The one-frequency dynamics of each locus of the UGA U^{f_1} over the first 500 generations of a single run . (If the animation does not work please download the full version of this manuscript from www.cs.brandeis.edu/~kekib/GAWorkings.html)
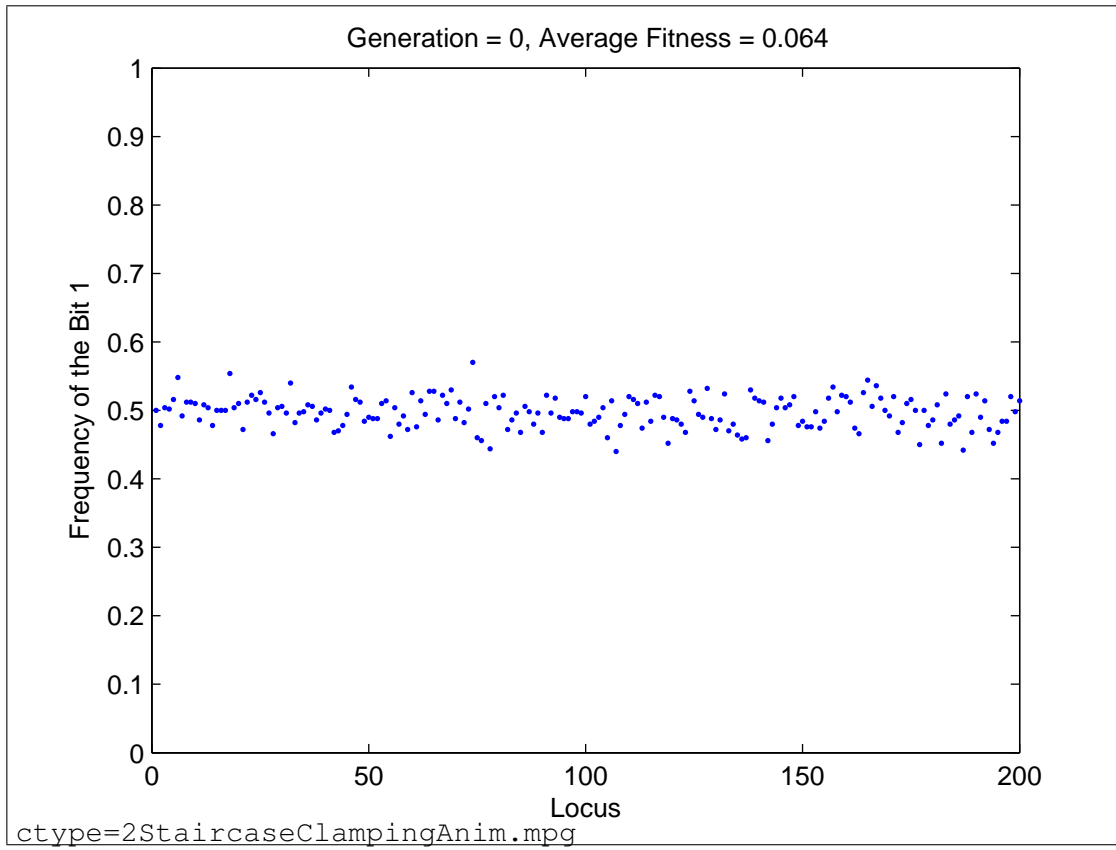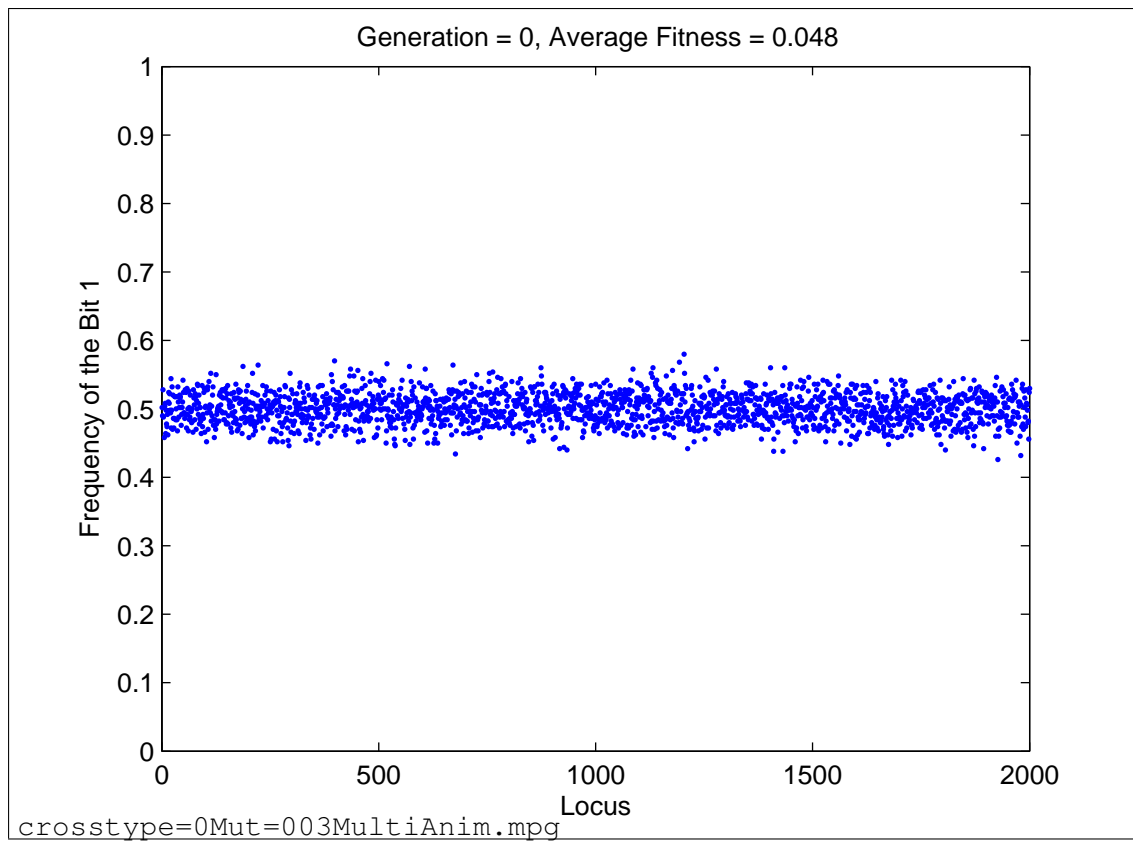
Animation 2: [Click on image to play] The one-frequency dynamics of each locus of the UGA $U_c^{f_1}$ over the first 500 generations of a single run. (If the animation does not work please download the full version of this manuscript from www.cs.brandeis.edu/~kekib/GAWorkings.html)
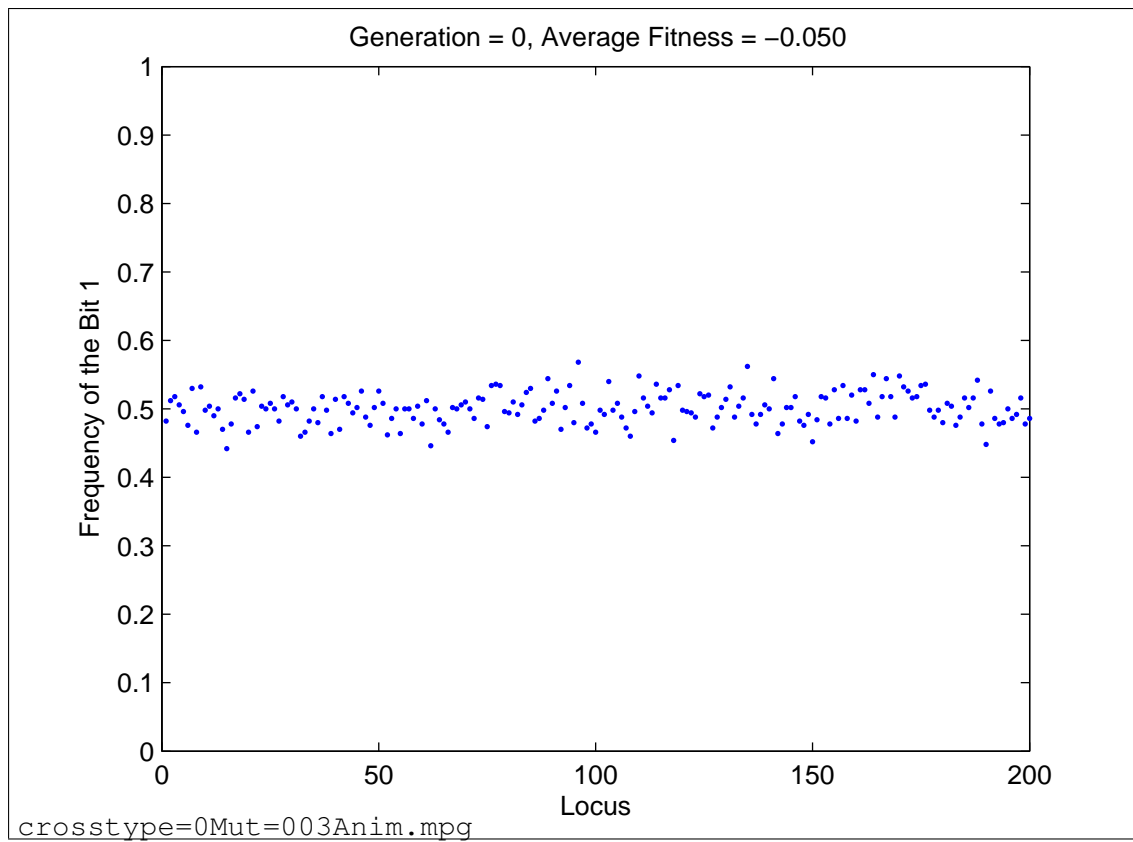
Animation 3: [Click on image to play] A visualization of the one-frequency dynamics of each locus over the first 500 generations of a single run of the MGA M^{f_1}. (If the animation does not work please download the full version of this manuscript from www.cs.brandeis.edu/~kekib/GAWorkings.html)

Animation 4: [Click on image to play] A visualization of the one-frequency dynamics of each locus over the first 500 generations of a single run of the UGA U^{f_2}. (If the animation does not work please download the full version of this manuscript from www.cs.brandeis.edu/~kekib/GAWorkings.html)

Animation 5: [Click on image to play] A visualization of the one-frequency dynamics of each locus over the first 500 generations of a single run of the MGA M^{f_2}. (If the animation does not work please download the full version of this manuscript from www.cs.brandeis.edu/~kekib/GAWorkings.html)