On Favoring Positive Correlations between Form and Quality of Candidate Solutions via the Emergence of Genomic Self-Similarity

Ivan Garibay
School of Computer Science
and Office of Research
University of Central Florida
12443 Research Pkwy,
Suite 302,
Orlando, FL 32826-3252
igaribay@cs.ucf.edu

Annie S. Wu
School of Computer Science
University of Central Florida
P. O. Box 162362
Orlando, FL 32816-2362
aswu@cs.ucf.edu

Ozlem Garibay
School of Computer Science
University of Central Florida
P. O. Box 162362
Orlando, FL 32816-2362
ozlem@cs.ucf.edu

ABSTRACT

A key property for the effectiveness of stochastic search techniques, including evolutionary algorithms, is the existence of a positive correlation between the form and the quality of candidate solutions. In this paper, we show that when the ordering of genomic symbols in a genetic algorithm is completely independent of the fitness function and therefore free to evolve along the candidate solutions it encodes, the resulting genomes self-organize into self-similar structures that favor this key stochastic search property.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: General

General Terms

Algorithms

Keywords

genetic algorithm, representation, proportional genetic algorithm, self-organization, genomic self-similarity, emergence

1. INTRODUCTION

In this paper, we use a Proportional Genetic Algorithm (PGA) [33], to study the emergent ordering of genomic symbols in the complete absence of selective pressure for a particular order. We hypothesize that self-similarity emerges because self-similar genomes are more robust with respect to crossover and mutation and because it favors positive correlations between the form and quality of candidate solutions. The PGA is a Genetic Algorithm (GA) [17, 12] with a representation based on protein concentrations rather than on the usual gene ordering. A PGA translates strings of genes into multisets of proteins prior to fitness evaluation. As a result, there is no fitness pressure for any particular gene ordering and the order of the genes is free to evolve along with the candidate solutions that they encode. Previous studies [33] have shown that genomic symbols under these circumstances are evenly distributed throughout the genome and that they appear to form building blocks of a peculiar type: coarse grained versions of the entire genome. In this paper, we ask the fundamental question: *what is the emergent genomic ordering when there is no selective pressure for any particular ordering?*. We use two very different methods to analyse the emergent genomic structure: equal-symbol correlation analysis, originally proposed by Voss [32] to analyse DNA structure, and an experimental method of our own making to analyse the self-similarity of genomic segments with repect to fitness. Our results can be summarized as follows:

1. The equal-symbol correlation on completely location independent genomes, as implemented by the PGA, resembles white noise behavior and the emergent genomic structure is self-similar with respect to fitness.

2. Emergent genomic self-similarity seems to produce the following effect: it favors positive correlations between form and quality of candidate solutions, a key property needed for stochastic search algorithms such as evolutionary algorithms; and it reduces schemata disruption caused by crossover.

2. BACKGROUND

2.1 The Proteomic Approach to Evolutionary Computation

Proteins are the basic building blocks of life. Evolutionary computation (EC) traditionally relies on gene analogous structures to represent candidate solutions for a given problem. In [10] Garibay investigates whether self-organization of protein analogous structures at the representation level can increase the degree of complexity and "novelty" of solutions obtainable using evolutionary search techniques. In

GECCO'05, June 25–29, 2005, Washington, DC, USA.

this proteome-based representational approach, functional structures analogous to proteins act as complexity builders for a genome. As in many representational approaches, the proteome-based approach is motivated by the observation that nature seems to compress or at least bias the information encoded in a genome. The genome determines the size of the search space and it is the target of evolutionary changes. It is, however, the uncompressed version of the genes – the organism – that is tested for fitness in a given environment. Furthermore, recent biological evidence suggests that the complexity of an organism is not necessarily correlated with the number of genes in their genome but more with the process of gene expression. For instance, there are more genes in the rice genome than in the human genome; and differences in the brain composition among humans and other primates are not so much due to differences in the genetic make up but to differences in amounts of expressed genes—proteins.

The biological processes motivating many representational approaches such embryogenesis and development, are far from understood, but there are some points that are widely accepted. The "central dogma" of molecular biology is one of them. The central dogma, in essence, says that DNA is translated into proteins and that proteins, not genes, are the basic building blocks of biological functionality. In the proteomic approach, we use the central dogma as a guiding principle. We represent solutions not using a genome, as classic EC does, but base our problem representation on the protein complement of the genome—the proteome.

There are two fundamental aspects introduced by a proteome-based representation: (1) proteins interact in a three dimensional medium analogous to a soup or multiset; and (2) proteins are functional structures. The first aspect is foundational for the study of the second. The PGA focuses on analyzing the effects on EC introduced by the first aspect: using a "content" oriented structure such as a multiset of proteins instead of a traditional "order" oriented structure such as a string of genes as the basis for problem representation. In this representational approach, the genetic operators act upon traditional strings of genes associated with the multisets of proteins. As a result, genetic operators are unchanged from traditional GA operators. We call these types of representations *completely location independent* or *proteome-based location independent* representations. For a complete description of this approach and completely location independent representations, we refer the reader to [10].

2.2 Proportional Genetic Algorithm

The Proportional Genetic Algorithm is a GA with a representation inspired by the genome to proteome mapping. In a GA, an individual is typically represented as a binary string. A population of these strings undergoes continuous genetic variation and fitness-based selection to produce more fit individuals. In the PGA, individuals are strings over a multicharacter alphabet and they undergo the same genetic variation. The difference is that fitness is not evaluated on these strings but on their associated multisets. The multiset associated with a string of symbols is simply the multiset containing all and only the elements in the corresponding string. For instance, the associated multiset of the string "aab" is $\{|a, a, b|\}$. Note that the strings "aab", "aba", and "baa" all have the same associated multiset and are thus indistinguishable with respect to fitness. Consequently, there is no pressure to select any particular ordering of two "a"s and one "b" in a genome. For a full description of the PGA and its applications, we refer the reader to [33, 34].

2.3 Related Work

The importance of the arrangement of encoded information has been recognized since the introduction of the GA. Positional bias [7], which occurs in typical location dependent representations, refers to the fact that encoded information that are relatively far apart on an individual are more likely to be disrupted by crossover than encoded information that are close together. As a result, how information is arranged in a problem representation can affect the evolvability of a GA.

A number of approaches have been investigated to deal with this problem, including operators such as inversion [17, 12] and uniform crossover [31] which have little or no positional bias. A significant amount of research has also been devoted to examining ways in which problem representation may be designed to improve the GA search process.

Numerous studies have focused on representations that allow a GA to dynamically evolve the arrangement of information on an individual including the arrangement of individual characters [1, 9, 14, 13, 16, 21] and the arrangement of groups of characters [5, 27, 30, 35]. While these approaches allow for dynamic evolution of the organization of the encoded information, the mapping from a GA individual to a problem solution still retains some aspects of order: (1) the order in which characters appear in an individual may affect their expression, and (2) some sort of reordering is typically required to decode an individual into a solution. A side effect of the latter is that each character or group of characters has a specific meaning or use, e.g. the 2nd binary digit or the 4th parameter. As a result, all encoding characters or groups must exist in order for a solution to be formed.

Purely content based problem representations have been developed in the form of problem specific structures such as condition/action rules [15, 36] and other problem specific elements, for example, [8]. These representations do not have a pre-defined set of required components. All encoded information is used to form a solution; non-existing information is not used.

There is recent interest in representational approaches that increase the complexity of solutions without increasing the complexity of genomes in order to better scale evolutionary search. These approaches share some degree of implicit self-organization. For instance the concepts of developmental mappings [23, 24, 22], implicit embryogenies [26, 3], developmental biology [4, 25],generative representations [19, 20], molecular computing [6], self-replicating sequences [11, 2], cellular automata variations [28, 29], and constrained generating procedures [18] all share, to some degree, this perspective in which some sort of self-organizing principle drives the representation of the problems.

3. EMERGENT WHITE NOISE BEHAVIOR

It has been shown using standard spectral density measurement techniques that individual base positions in DNA sequences exhibit $1/f$ noise and long-range fractal correlations [32]. From the evolutionary computation perspective, we pose the following question: What kind of behavior do genomic symbols exhibit in evolutionary computation individuals? After a brief analysis, it is easy to realize that for

most evolutionary computation representations, which rely on an order-based encoding of information, this question has a trivial answer: the behavior exhibited by the evolved symbolic sequences is the one dictated by the chosen fitness function. For proteome-based location independent representations such as the PGA, this question becomes more interesting and less obvious. In proteome-based location independent representations, the encoding is based on which symbols are present on a genome and not on the order in which they are present [33]. Therefore, the order of genomic symbols is free to evolve along with the candidate solution they encode. We use spectral density measurement techniques to analyze the best individuals obtained by a PGA when applied to three problem domains: number matching, symbolic regression, and dynamical system control.

3.1 Symbolic Sequence Analysis

Autocorrelation and *spectral density* functions are widely used to analyze how fluctuations of a quantity, $X(t)$, are correlated between times t and $t+\tau$. The autocorrelation function, $C(\tau)$, is a quantitative measure of the correlations in the fluctuations in $X(t)$:

$$C_X(\tau) = \langle X(t) \times X(t+\tau) \rangle$$

where the brackets denote sample averages. The spectral density function, $S_X(f)$, provides additional correlation information using the Fourier coefficient of $X(t)$ at frequency f:

$$X(f) \propto \int X(t) e^{2\pi f t} dt$$

then

$$S_X(f) = \frac{|X(f)|^2}{\Delta f}$$

where the effective bandwidth of the Fourier integral is Δf. The autocorrelation and spectral density functions are related by the following equations:

$$S_X(f) \propto \int C_X(\tau) \ \texttt{cos}\ 2\pi\tau \ d\tau$$

$$C_X(\tau) \propto \int S_X(f) \ \texttt{cos}\ 2\pi f \ df.$$

Figure 1 shows examples of typical noises and their characteristic spectral densities $S(f)$.

Voss [32] has shown that this time correlation analysis can be successfully adapted for analysis of symbolic sequences such DNA. DNA consists of sequences of $K = 4$ bases or symbols. Given a symbolic sequence of length N, $X_n, n = 1, 2, ..., N$, where each X_n is one of the symbols $k, k = 1, 2, ..., K$ he first defines the following *equal-symbol multiplication* function,

$$X_n \times X_m = \begin{cases} 1 & if X_n = X_m \\ 0 & otherwise. \end{cases}$$

Voss then isolates a binary sequence for each symbol using $U_k[X_n]$ which identifies the occurrences of the symbol k in a sequence X_n.

$$U_k[X_n] = \begin{cases} 1 & if X_n = k \\ 0 & otherwise \end{cases}$$

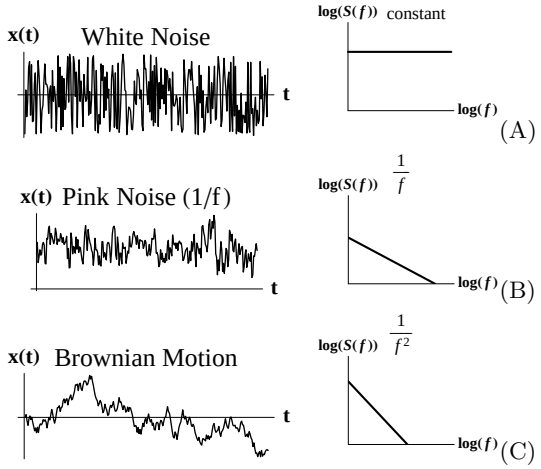

Figure 1: White(A), Pink(B) and Brownian Motion(C) noises and their spectral densities, S(f).

He defines

$$X_n \times X_m = \sum_{k=1}^{K} U_k[X_n] U_k[X_m]$$

which leads to the calculation of the equal-symbol autocorrelation function for each symbol k

$$C(\tau) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} U_k[X_n] U_k[X_{n+\tau}] = \sum_{k=1}^{K} C_k(\tau).$$

Given that $X_n = k$, $C_k(\tau)$ gives the probability that $X_{n+\tau} = k$ is also true. The spectral density is calculated using the following equation

$$S(f) = \sum_{k=1}^{K} S_k(f)$$

where $S_k(f)$ is the equal-symbol spectral density of symbol k.

We use standard spectral density measurement techniques as adapted by Voss to analyze GA genomes using proportional representation. We analyze the best genomes from the last generation of a PGA run. These individuals are likely to be optimal or very close to optimal. Figure 2 shows the spectral density of the first two symbols of the alphabet (A and B) for three problems: number matching, symbolic regression, and control of dynamical systems. In all cases, we find that the emergent ordering of genomic symbols, according to our symbolic sequence analysis, resembles white noise. These results contrast with the original analysis of DNA sequences from GenBank data bank. In that study, $1/f$ noise and long-range fractal correlations are found over a broad range of DNA classifications such primate, invertebrate, plant, etc.

Regarding this contrast between the PGA and DNA spectral analysis, evolutionary computation processes are inspired by nature but do not pretend to model the complexities involved in biological evolution. Therefore, PGA is not a model to study DNA. Having said so, we think that the observed differences are due to the fact that DNA strings are

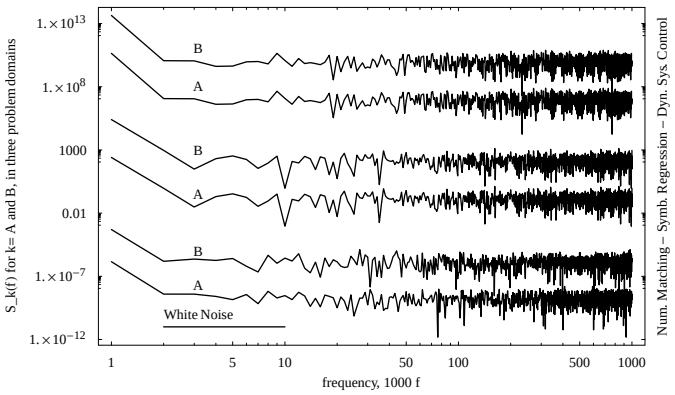

Figure 2: Log-log spectral density plots of the best of the last generation individuals. From the bottom up, symbols A and B for PGA on number matching, on symbolic regression and on dynamical system control problems. The graphics have been offset for clarity.

a mixture of location dependent and location independent encodings, while the PGA is purely location independent. Since the PGA is purely location independent, we expect that the emerging order is purely due to the dynamics of the algorithm, in our case white noise. In the DNA case, it seems that the pink noise arises due to long range dependencies in the encoded information.

4. GENOMIC SELF-SIMILARITY ANALYSIS

The objective of this empirical study is to determine whether or not genomic self-similarity with respect to fitness emerges in the PGA proteome-based location independent representation. Based on previous PGA studies that suggest that PGA genome segments are coarse grained versions of an entire PGA genome, we expect the following behavior. Large genome segments will approximate the fitness of the entire genome very closely. The smaller the genome segments, the less they will resemble the fitness of the entire genome. There will be a cut-off point at which a segment is too small to represent the required information and self-similarity is lost. As a result, we expect a gradual decrease in fitness as the segments get smaller until a cut-off point is reached at which point fitness will show a significant drop, as shown in Figure 3(C). As a baseline comparison, we use a regular GA in which genomic order is dictated by selection pressure and consequently not free evolve or self-organize. Exceeding our expectations, and as we will see shortly, PGA genomes resemble ideal self-similarity until the cut-off point is reached.

The freedom of genomic elements to self-organize comes from the choice of representation. In the PGA proteome-based location independent representation, the order in which the symbols are arranged on a genome, by definition, has no effect on the fitness of the individual. As a result, the genome is free to organize in response to other factors such the dynamics of the algorithm, operators, and building block processing. Because the PGA representation is based on a multiset, we can guarantee that no fitness function definition can distinguish between different orderings of the same symbols.

On the other hand, in a traditional representation where information is encoded in the order of the genomic symbols, there is no such freedom for genomic self-organization. In this case, different orderings of symbols typically represent different solutions; consequently, the ordering of symbols is subject to selection pressure. Unless a fitness function is specifically handcrafted so that all possible orderings of the same multiset of symbols have the same fitness value, a very unlikely scenario, the ordering of symbols will be determined solely by the fitness function. In our experiments, we do not handcraft such fitness for the traditional GA binary representation; hence, we do not expect self-similarity to emerge on GA genomes.

4.1 Self-Similarity Metric for Genomes

We use fitness as the metric for genomic self-similarity. A genome is self-similar if its fitness is approximately equal to the average fitness obtained by evaluating all of its genomic segments of a given segment length. Let us introduce the following notation. A genome $w = w_1 w_2 w_3 ... w_L$, is a string of length L over the genome alphabet Σ. A segment $s_{i,j} = s_i s_{i+1} s_{i+2} ... s_j$ of w is defined as the substring $w_i w_{i+1} w_{i+2} ... w_j$, where $1 \leq i \leq j \leq L$. A fitness function $f(w)$ maps strings into real numbers and is defined for any genome or segment. Using the notation above, we define the average fitness of all segments of size r over genome w as follows:

$$\hat{f}_r(w) = \frac{\sum_{i=1}^{L-r+1} f(s_{i,i+r-1})}{(L-r+1)} \quad (1)$$

Note that for a segment equal to the entire genome ($r = L$), the expression above reduces simply to a fitness evaluation:

$$\hat{f}_r(w) = f(s_{1,L}) = f(w)$$

Finally, we say that a genome is self-similar if the following expression is true:

$$\forall_r \left[\hat{f}_r(w) \approxeq f(w) \right] \quad (2)$$

where r is the size of the genome segments used to analyze self-similarity.

The above equation implies the following. For an ideal case of genomic self-similarity, Equation 2 will hold indefinitely for any segment size r. In this case, genomic segments of any size will have the same fitness as the entire genome, as shown in Figure 3(A). This case is analogous to perfect fractal behavior. On the other hand, if there is no self-similarity, Equation 2 will not hold even for large segments. In this case, the fitness of the segments will not resemble the fitness of the entire genome. We thus expect random segments with average fitness equal to the median fitness of the problem as demonstrated in Figure 3(B).

4.2 Fitness Evaluation

For all experiments, we set the algorithms to solve a very simple problem: *number matching*. Number matching is a simple hamming distance type of problem for real numbers. The goal is to match, as closely as possible, a vector of target numbers. We use vectors of size two. For convenience of

Figure 3: Predicted genomic self-similarity in populations that have converged to the optimal fitness of 1. (A) Ideal self-similar case: all genomic segments have fitness of 1 regardless of their size. (B) No self-similarity case: segments have random fitness with average equal to the fitness median. (C) PGA expected self-similarity: segments gradually reduce in fitness as they get smaller. Experimental results resemble the ideal case (see Figure 4).

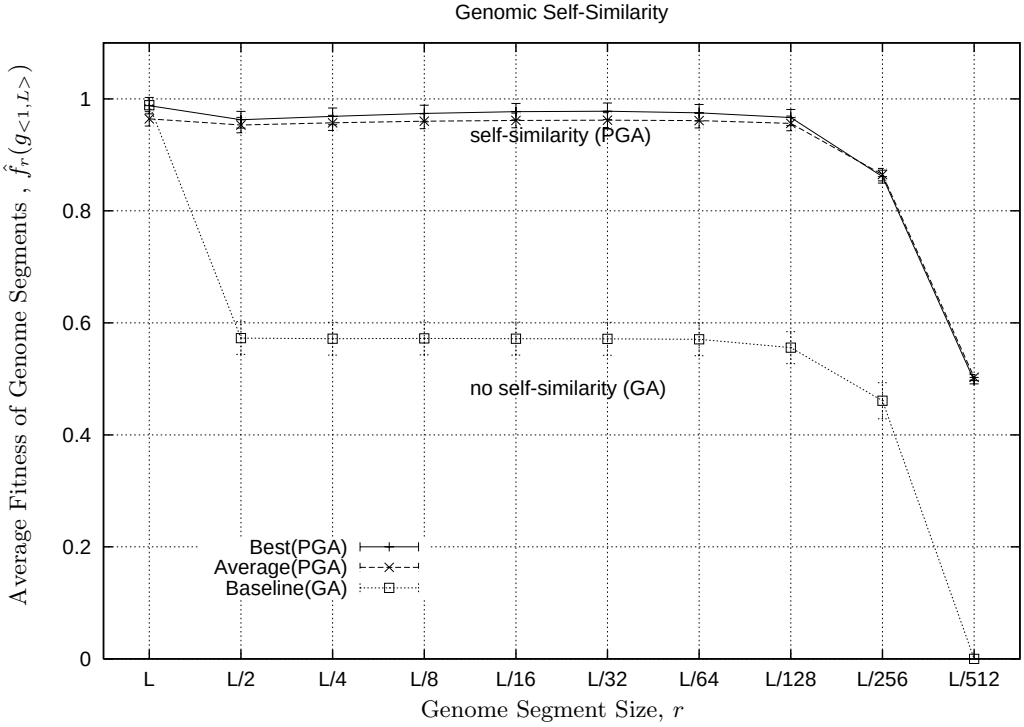

Figure 4: PGA best, PGA average, and GA baseline comparison of average fitness of genome segments($\hat{f}_r(g_{<1,L>})$) of sizes(r) L(entire genome), L/2(half genome), L/4, ... , and L/512 on a simple number matching problem averaged over 100 runs with 95% confidence intervals. (Top: self-similarity) PGA segment fitness remains approximately equal to the entire genome fitness for segment sizes of L/2, L/4, ... until L/128. (Middle: no self-similarity) No GA segment has average fitness close to the entire genome fitness for any segment size. Baseline note: for clarity, only GA average is shown; GA best is not shown but has similar behavior.

exposition, let us define the following function:

$$\delta(x,y) = \begin{cases} x/y & \text{if } x < y \\ y/x & \text{otherwise} \end{cases} \qquad \forall x,y \in \Re^+.$$

Fitness is determined by the average difference, as measured by δ, between the values encoded in the individual's genome (or segment) and the target values:

$$f(g_{<1,L>}) = \frac{\delta(v^1_{target}, v^1_{encoded}) + \delta(v^2_{target}, v^2_{encoded})}{2}$$

Where, $V_{target} = \{v^1_{target}, v^2_{target}\}$ is the vector of target values, and $V_{encoded} = \{v^1_{encoded}, v^2_{encoded}\}$ is the vector of values encoded in w. For the PGA, the encoded values are given by:

$$V_{encoded}(w) = \left[\frac{|w|_a}{|w|_a + |w|_b}, \frac{|w|_c}{|w|_c + |w|_d} \right] \qquad (3)$$

where the genomic alphabet is $\Sigma_{PGA} = \{a, b, c, d\}$, and $|X|_y$ returns the number of times a symbol y appears in the string X. Note that this definition is also valid for genomic segments. Note also that only the number of occurrences of symbols are used to calculate the values. As shown on Equation 3, the first value is encoded simply as a proportion between symbols a and b, and the second value, between symbols c and d. The proportions are numbers between zero and one, but they can be easily scaled to represent any parameter range of values.

For the GA, the values encoded in the genome are interpreted in the usual way. The genomic alphabet is binary: $\Sigma_{GA} = \{0, 1\}$. The first half of the genome represents our first encoded value and the second half, our second encoded value. Note that this definition is also valid for genome segments.

4.3 Settings

For our experiments, we use a standard GA and a PGA. The GA uses a binary alphabet; the PGA uses a multi-character alphabet. Mutation, in the GA, is bit-flip mutation. The PGA mutation, randomly changes one alphabet symbol into another. The following parameter settings are common for all experiments: the genome length is L=1000 and segment sizes are L/2, L/4, L/8, L/16, L/32, L/64, L/128, L/256, and L/512, the crossover type is two-point, the crossover rate is 0.8, the mutation rate is 0.005, the selection method is tournament of size 4, the population size is 250, and the number of generations is 500. We perform 100 trials for all experiments using new randomly generated targets for each run, and report average values with their 95% confidence intervals.

4.4 Results

Figure 4 shows plots of the average measured fitness for decreasing segment length. These results reveal genomic self-similarity with respect to fitness for the PGA and no self-similarity for the baseline case. The self-similarity is almost ideal (see Figure 3 (A)) for segments as small as $L/128 = 1000/128 = 7.81$ symbols. After this critical point, self-similarity is lost and fitness of the segments decreases significantly. Figure 5 shows the raw fitness of segments of size L/2 for the two encoded values over the 500 generations. In Figure 5 (A), the variance for the self-similar case is very small. This result indicates that all segments, not just their averages, must have fitness similar to the entire genome. In Figure 5 (B), the variance for the non-self-similar case is very high, indicating that the segments have a wide range of fitness values and are not similar to the entire genome.

5. DISCUSSION

Empirical observations from previous studies [33] and from this paper indicate that, during the course of the evolutionary process, PGA genomes tend to self-organize into a self-similar state with respect to fitness. In this state, alphabet symbols are evenly distributed throughout the genome. The question, then, is: *why does this particular organization emerges when there is no fitness pressure for any particular ordering?*

Self-similarity with respect to fitness is possible because PGA looks for proportions. If the desired proportion of symbols α and β is $2:1$, and if the symbols are evenly distributed throughout the genome, the genomic segments will have roughly the correct proportion. This self-similarity reduces building block disrruption. Building block disruption occurs when a crossover operation destroys useful schemata. In the PGA case, a genome self-organizes into self-similar building blocks. These building blocks are not disrupted by typical crossover operations because a sub-segment is itself a representative of the entire building block or schema. As a result, crossover disruption is minimized, if not eliminated, on the PGA genomes once this self-similarity has emerged.

Self-similarity reinforce positive correlations between form and quality of candidate solutions. Let us analyze the effects of self-similar genomes on one-point crossover and single-symbol mutation. Assume that P_1 and P_2 are two individuals with perfect, self-similar genomes of length L over the alphabet $A = \{a, b\}$. P_1 and P_2 undergo one-point crossover at location l and, as a result, the individuals P_3 and P_4 are produced. The concentrations of symbols a and b on individuals P_1 and P_2 are given by:

$$a_{p1} = \frac{|P_1|_a}{L} \qquad b_{p1} = \frac{|P_1|_b}{L}$$

$$a_{p2} = \frac{|P_2|_a}{L} \qquad b_{p2} = \frac{|P_2|_b}{L}$$

It is easy to see that after crossover, the concentrations of the resulting offspring P_3 and P_4 are:

$$a_{p3} = \frac{\frac{|P_1|_a}{L} \times l + \frac{|P_2|_a}{L} \times (L-l)}{L}$$

$$a_{p4} = \frac{\frac{|P_1|_a}{L} \times (L-l) + \frac{|P_2|_a}{L} \times l}{L}$$

$$b_{p3} = \frac{\frac{|P_1|_b}{L} \times l + \frac{|P_2|_b}{L} \times (L-l)}{L}$$

$$b_{p4} = \frac{\frac{|P_1|_b}{L} \times (L-l) + \frac{|P_2|_b}{L} \times l}{L}$$

Hence, the symbol concentration of the offspring produced as a result of one-point crossover between parents with self-similar genomes is the weighted average of the symbol concentration of the parents. In terms of mutation, let us assume that P_1 undergoes single-symbol mutation in the following way: a single symbol is changed from a to b. Before mutation, the concentrations of symbols in P_1 are given

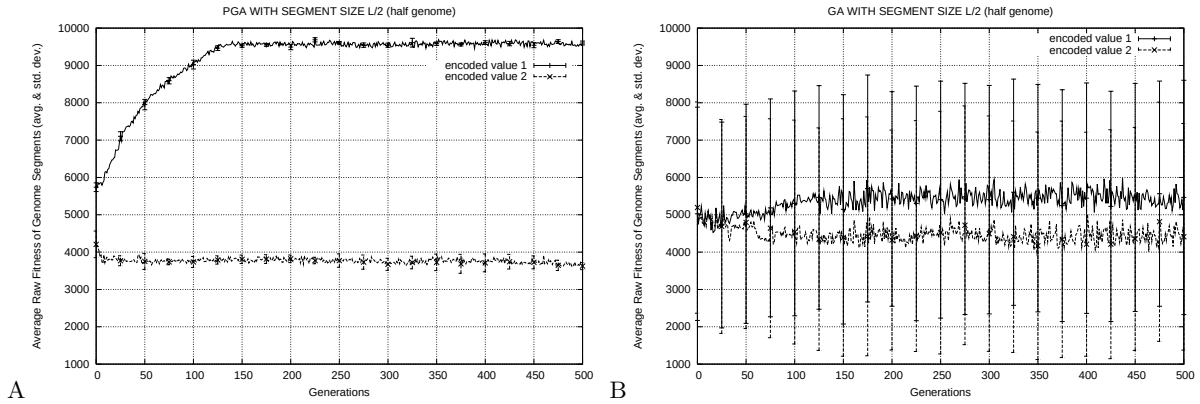

Figure 5: Average encoded values 1 and 2 of genomic segments of size L/2 (half-genome) over 500 generations. The plots show averages and standard deviations over 100 runs. (A) PGA encoded values 1 and 2 have tight standard deviations that indicate that encoded values of PGA genomic segments of size L/2 converge. From Figure 4 we observe that their fitness converges to a near optimal solution, similar to the fitness of the entire genome L. (B) GA encoded values 1 and 2 have wide standard deviations that indicate that GA genomic segments of size L/2 do not converge. Furthermore, the encoded values of these segments appear to be randomly distributed. ¿From Figure 4 we observe that their average normalized fitness is not similar to the entire genome, but near to the median 0.5 instead. We observe similar behavior for segments of size L/4 to L/128.

above. After mutation, we have:

$$a_{p1}^{'} = \frac{|P_1|_a - 1}{L} \qquad b_{p1}^{'} = \frac{|P_1|_b + 1}{L}$$

The change in concentrations is given by:

$$\Delta_M a_{p1} = |\frac{|P_1|_a}{L} - \frac{|P_1|_a - 1}{L}| = \frac{1}{L}$$

$$\Delta_M b_{p1} = |\frac{|P_1|_b}{L} - \frac{|P_1|_b + 1}{L}| = \frac{1}{L}$$

This concentration change equals the minimum possible change allowed by the encoding: $1/L$.

Clearly, the behavior of crossover and mutation applied to self-similar genomes produces smooth moves in the representation space. Crossover acts by averaging parent concentrations and mutation acts by changing an individual by the minimal difference. This smoothness of the proteome-based location independent representation will certainly not hinder, and, according with the previous analysis, will accentuate positive correlations between form and quality of candidate solutions.

6. CONCLUSIONS

This paper studies the emergent organization of proteome-based location independent genomes using two methods: standard spectral density analysis of equal-symbol correlations and an empirical fitness comparison of genomic segments to determine self-similarity with respect to fitness. The equal-symbol correlation analysis shows that the order of genomic symbols resembles white noise. These results confirm previous results [33] that show that in PGA genomes, symbols are evenly distributed throughout the entire genome. The empirical analysis offers evidence of the emergence of genomic self-similarity with respect to fitness in PGA genomes. When genomic order is free to evolve, the genome self-organizes in response to the dynamics of the evolutionary system. In the PGA case, it self-organizes into a fractal-like structure in which genomic segments have approximately the same fitness as the entire genome. These fractal-like genomic structures appear to favor a key property for stochastic search: the positive correlation between form and quality of candidate solutions.

7. ACKNOWLEDGEMENTS

We thank Ken De Jong for comments on an early version of this paper and the members of the UCF Evolutionary Computation Laboratory for useful discussions.

8. REFERENCES

[1] J. D. Bagley. *The behavior of adaptive systems which employ genetic and correlation algorithms*. PhD thesis, University of Michigan, 1967.

[2] W. Banzhaf, P. Dittrich, and B. Eller. Selforganization in a system of binary strings with topological interactions. *Physica D*, 125:85–104, 1999.

[3] P. J. Bentley. Natural design by computer. In *Proceedings of the 2003 AAAI Spring Symposioum: Computational Synthesis: From Basic Building Blocks to High Level Functionality*, pages 1–2. American Association for Artificial Intelligence, AAAI Press, 2003. Technical Report SS-03-02.

[4] P. J. Bentley. Fractal proteins. *Genetic Programming and Evolvable Machines Journal*, 5:71–101, 2004.

[5] D. S. Burke, K. A. De Jong, J. J. Grefenstette, C. L. Ramsey, and A. S. Wu. Putting more genetics into genetic algorithms. *Evolutionary Computation*, 6(4):387–410, 1998.

[6] M. Conrad. Computation: Evolutionary, neural, molecular. In *2000 IEEE Symposium on Combinations*

of Evolutionary Computation and Neural Networks, pages 1–9, 2000.

[7] L. J. Eshelman, R. A. Caruana, and J. D. Schaffer. Biases in the crossover landscape. In J. D. Schaffer, editor, *Proc. 3rd Int'l Conference on Genetic Algorithms*, pages 10–19, 1989.

[8] R. W. Franceschini, A. S. Wu, and A. Mukherjee. Computational strategies for disaggregation. In *Proc. 9th Conf. on Computer Generated Forces and Behavioral Representation*, 2000.

[9] D. R. Frantz. *Non-linearities in genetic adaptive search*. PhD thesis, University of Michigan, 1972.

[10] I. Garibay. *The Proteomics Approach to Evolutionary Computation: An Analysis of Proteome-Based Location Independent Representations based on the Proportional Genetic Algorithm*. PhD thesis, University of Central Florida, 2004.

[11] M. Garzon, D. Blain, K. Bobba, A. Neel, and M. West. Self-assembly of DNA-like structures in silico. *Genetic Programming and Evolvable Machines*, 4(2):185–200, 2003.

[12] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison Wesley, 1989.

[13] D. E. Goldberg, K. Deb, H. Kargupta, and G. Harik. Rapid accurate optimization of difficult problems using fast messy genetic algorithms. In S. Forrest, editor, *Proc. 5th Int'l Conference on Genetic Algorithms*, pages 56–64, 1993.

[14] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3:493–530, 1989.

[15] J. Grefenstette, C. L. Ramsey, and A. C. Schultz. Learning sequential decision rules using simulation models and competition. *Machine Learning*, 5(4):355–381, 1990.

[16] G. R. Harik. *Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms*. PhD thesis, University of Michigan, 1997.

[17] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[18] J. H. Holland. *Emergence: From Chaos to Order*. Addison-Wesley, 1998.

[19] G. S. Hornby, H. Lipson, and J. B. Pollack. Generative representations for the automated design of modular physical robots. *IEEE Transactions on Robotics and Automation*, 2003. In Press.

[20] G. S. Hornby and J. B. Pollack. Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, 8(3):223–246, 2002.

[21] H. Kargupta. The gene expression messy genetic algorithm. In *Proc. IEEE Int'l Conference on Evolutionary Computation*, pages 814–819. IEEE Press, 1996.

[22] J. R. Koza et al. Automated synthesis by means of genetic programming of human-competitive designs employing reuse, hierarchies, modularities, development, and parameterized topologies. In *Proceedings of the 2003 AAAI Spring Symposioum: Computational Synthesis*, pages 138–145. AAAI Press, 2003.

[23] J. R. Koza, F. H. B. III, D. Andre, and M. A. Keane. *Genetic Programming III*. Morgan Kaufmann Publishers, 1999.

[24] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, 2003. ISBN 1-4020-7446-8.

[25] S. Kumar. Multicellular development, self-organization, and differentiation. In *Proc. GECCO 2004 Workshop on Self-organization in Representations for Evolutionary Algorithms: Building complexity from simplicity*, 2004.

[26] S. Kumar and P. J. Bentley, editors. *On Growth, Form and Computers*. Academic Press, 2003.

[27] H. A. Mayer. ptGAs–genetic algorithms evolving noncoding segments by means of promoter/terminator sequences. *Evolutionary Computation*, 6(4):361–386, 1998.

[28] J. Miller and P. Thomson. Beyond the complexity ceiling: Evolution, emergence and regeneration. In *Proc. GECCO 2004 Workshop on Regeneration and Learning in Developmental Systems*, 2004.

[29] L. M. Rocha. Evolving memory: Logical tasks for cellular automata. In *Proc. Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9)*. In Press, 2004.

[30] T. Soule and A. E. Ball. A genetic algorithm with multiple reading frames. In L. Spector, E. D. Goodman, A. S. Wu, W. B. Langdon, H. M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proc. Genetic and Evolutionary Computation Conference*, 2001.

[31] G. Syswerda. Uniform crossover in genetic algorithms. In *Proc. 3rd Int'l Conference on Genetic Algorithms*, 1989.

[32] R. F. Voss. 1/f noise and fractals in DNA-base sequences. In Crilly, Earnshaw, and Jones, editors, *Applications of Fractals and Chaos*, pages 7–20. Springer-Verlag, 1993.

[33] A. S. Wu and I. Garibay. The proportional genetic algorithm: Gene expression in a genetic algorithm. *Genetic Programming and Evolvable Hardware*, 3(2):157–192, June 2002.

[34] A. S. Wu and I. Garibay. Intelligent automated control of life support systems using proportional representations. *IEEE Transactions on Systems, Man, and Cybernetics—Part B*, 34(3):1423–1434, June 2004.

[35] A. S. Wu and R. K. Lindsay. A comparison of the fixed and floating building block representation in the genetic algorithm. *Evolutionary Computation*, 4(2):169–193, 1996.

[36] A. S. Wu, A. C. Schultz, and A. Agah. Evolving control for distributed micro air vehicles. In *Proc. IEEE Int'l Symp. Computational Intelligence in Robotics and Automation*, 1999.