Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 1 (2012) 1005–1014

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2010

Design of a dynamic model of genes with multiple autonomous regulatory modules by evolutionary computations

Alexander V. Spirov[a*] and David M. Holloway[b]

a) State University of New York at Stony Brook, Computer Science Department and Center of Excellence in Wireless & Information Technology, Stony Brook University Research & Development Park, 1500 Stony Brook Road, Stony Brook, NY 11794-6040, USA
b)Mathematics Department, British Columbia Institute of Technology, Burnaby, B.C., Canada; Biology Department, University of Victoria, B.C., Canada.

Abstract

A new approach to design a dynamic model of genes with multiple autonomous regulatory modules by evolutionary computations is proposed. The approach is based on Genetic Algorithms (GA), with new crossover operators especially designed for these purposes. The new operators use local homology between parental strings to preserve building blocks found by the algorithm. The approach exploits the subbasin-portal architecture of the fitness functions suitable for this kind of evolutionary modeling. This architecture is significant for Royal Road class fitness functions. Two real-life Systems Biology problems with such fitness functions are implemented here: evolution of the bacterial promoter rrnP1 and of the enhancer of the Drosophila even-skipped gene. The effectiveness of the approach compared to standard GA is demonstrated on several benchmark and real-life tasks.

Evolutionary computations; Genetic algorithms; Crossover operators; building blocks; dynamic models of genes; multiple autonomous regulatory elements.

1. Introduction

This paper describes an evolutionary approach used to generate functional gene regulatory networks, with the goal of testing new crossover algorithms inspired by nature. Recently, such approaches have become more widely used, as evolutionary simulations or evolution *in silico* [1;2;3;4;5;6;7]. Computer-simulated evolution is used to optimize kinetic models of genes with multiple regulatory elements (Cis-Regulatory Modules, CRMs). We describe the actions of upstream transcription factors on the genes and the structure of the CRMs in terms of the kinetics of gene expression (see [7;8] for details). These interactions create spatial expression patterns of the genes under study. Parameters describing the structure of the CRMs and interactions between the transcription factors and their targets (CRMs) are randomly mutated in an evolutionary procedure, and tested for the effect on gene expression. A parameter change which improves the fit between the gene expression model and the expression pattern observed

* Corresponding author. Tel.: +1-631-632-5582; fax: +1-631-632-4653.
E-mail address: alexander.spirov@gmail.com.

1877-0509
doi:10.1016/j.procs.2010.04.111

experimentally is retained, a general strategy in evolutionary computation.

Classical understanding of the mechanisms behind biological evolution served as inspiration for an entire order of heuristic optimization techniques, known generally as Evolutionary Computations (EC). In the past decade, research in molecular biology and genetics has conclusively shown that living organisms successfully utilize biomolecular implementations of EC for effective solution of problems in survival and adaptation. The most obvious examples of this type would be the mechanisms of antibody selection in a higher organism's adaptive immune system [9]; and in their counterparts, the mechanisms of antigen variability in pathogenic organisms, such as viruses and bacteria [10].

Natural GA (Genetic Algorithms) acts as a somewhat flexible hybrid optimization technique, used both in higher and lower organisms, albeit in differing ways. Specifically, our approach to EC is characterized by the use of operators that implement reproduction and diversification of genetic material in a manner inspired by the mechanisms of retroviral recombination [11] and the genetic-engineering technique known as DNA shuffling [12]. We refer to our overall technique as Retroviral Genetic Algorithms or retroGA.

RetroGA has many applications to problems of forced molecular evolution and has demonstrated impressive effectiveness on a series of benchmark tests [8]. We selected these tests on the basis of their potential similarity to real-world problems of in vitro evolution and molecular-biological evolution. We give special attention to the fitness functions as formal models, closely resembling real-world problems of molecular evolution. Some of the simplest fitness functions that demonstrate the properties of neutral subbasins linked by narrow pathways are the Royal Road (RR) and Royal Staircase (RS) functions.

Publications of van Nimwegen with co-authors [13;14;15;16;17] emphasized the population dynamics of various RR and RS fitness functions. van Nimwegen et al. also drew attention to RR functions as models of natural evolution. It is becoming clear that the dynamics of evolutionary processes on neutral fitness landscapes are qualitatively very different from evolutionary dynamics on rugged landscapes [13-16]. A major impetus for the present work is the lack of suitable models and theory for the landscapes encountered in real molecular evolution. Common conceptions of landscape structure (multi-modal or rugged) in the GA literature can be inapplicable for optimization in the class of evolutionary scenarios we deal with in this communication.

Here, we develop retroGA to the point where we can clearly test its utility in possible applications, such as directed in vitro molecular evolution and biomolecular computation.

2. The Approach

The molecular machines that rearrange DNA often process molecules according to certain signal sequences. From a computational point of view, these are analogous to marks or tags on a string. Molecular machines read these tags and interpret them as instructions for further string operations. Of the genetic diversification mechanisms that utilize such signal sequences, one of the most simple and well-known is retroviral recombination.

Retroviral Recombination: Recombination is the process by which progeny receive an arrangement of genes that is different from that of either parent [9;11].

Generalization of Retroviral Recombination: The techniques of DNA shuffling (Sex PCR in particular) and Random-Priming Recombination (RPR) [12; 18] can be generalized from retroviral recombination. (For biological details and basic references on retroviral recombination and Sex PCR see [8].)

2.1. The retroGA technique

Our technique is characterized by the use of operators that implement the reproduction and diversification of genetic material in a manner inspired by retroviral reproduction (the Reproduction/Crossover operator, RC) and a genetic-engineering technique known as DNA shuffling (the Generalized Replication/Crossover operator, GRC). These are our primary genetic operators, as alternatives to classical crossover and mutation operators. In everything else, our approach follows classical GA.

The Reproduction/Crossover operator: The RC operator generates a child string from a given parent pair, combining the functions of reproduction and crossover (Figure 1). Template switch (crossover) between the parent strings takes place in the regions of local homology of the strings (for details see [8]).

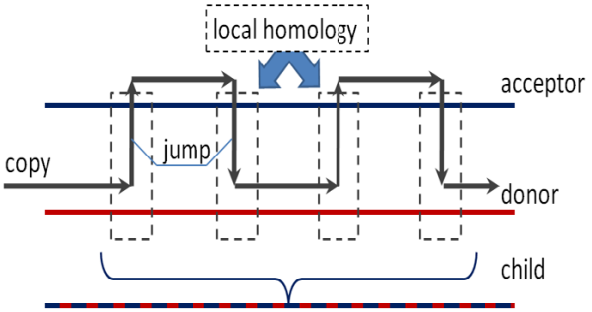

Figure 1 Template switching (crossover) in the RC operator.

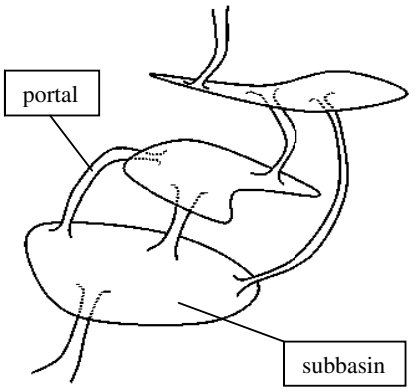

Figure 2 General idea of a subbasin and portal architecture for the Royal Road / Royal Staircase fitness functions. (After [14]).

Homology-based PCR techniques may be naturally interpreted as a generalization of retroviral recombination processes. This inspired us to develop a generalization of the RC operator - the Generalized Replication/Crossover operator (GRC operator). The operator uses not two but N parental strings to produce one child sequence (See [8] for further details).

2.2. Fitness functions to study hard evolutionary problems

In the current literature dealing with biological and artificial evolution, one can find various grades and classifications of difficult problems having to do with biology [19]. It is these benchmark problems that we will focus on below.

In many combinatorial optimization problems as well as in biological molecular evolution, we can use the "building block" (BB) hypothesis [20;21]. This states that a solution can be decomposed into a number of BBs, which can be searched for independently and afterwards be combined to obtain a good or even optimal solution. The remaining part of this paper is substantially based on the BB hypothesis.

Adaptive Landscapes: Wright's [22] creation of the "adaptive landscape" metaphor has had a strong effect on the theoretical analysis of evolutionary processes. The point of view that a typical combinatorial optimization and biological evolution fitness function may be modeled by rugged landscapes has gained considerable currency in recent years [23].

Subbasin-portal Architecture: At the same time as the idea of rugged landscapes was gaining momentum, an alternative concept was also being developed. This idea was based on the hypothesis of substantial degeneracy in the genotype-to-phenotype and the phenotype-to-fitness mappings. The history of this idea dates back to Kimura [24], who argued that on the genotypic level, most genetic variation occurring in evolution is adaptively neutral with respect to phenotype. During neutral evolution, different genotypes in a population fall into a relatively small number of distinct fitness classes, each consisting of a set of genotypes with approximately equal fitness. In other words, the genotype space decomposes into a set of subbasins of iso-fit genotypes that are entangled with each other in a complicated fashion (Figure 2). Through neutral mutations, individuals walk randomly in a given subbasin, until one of them finds a connection (or portal) to a subbasin of higher fitness.

2.3. Benchmark problems to simulate real molecular evolution

The aforementioned class of fitness functions with subbasin-portal architecture has already found a practical application in analyzing the evolution of the secondary structure of RNA [25;26]. Another example of molecular evolution with a predominance of neutral drift that has been widely discussed in the scientific literature is the evolution of the regulatory regions of genes. Several characteristics of regulatory regions, such as functional

redundancy, modular architecture, and low sequence specificity of transcription factors (TFs) for their binding sites (BS), suggest that they have freedom to change without drastic changes in their functions [27;28;29]. The commonly held conception is that most single mutations in regulatory regions represent a very small contribution to phenotypic variation, allowing the fixation of weak deleterious mutations by genetic drift [30;31]. Such an application of Kimura's ideas on the phenotypically close-to-neutral character of many (or the majority of) point mutations to the evolution of regulatory regions/modules may be naturally modeled by fitness functions that exhibit subbasin-portal architecture. A more general case, where a weak deleterious (maladaptive) mutation in a single site of a regulatory region is compensated by a mutation in a different region (compensatory neutral mutations [32]) is also assumed in the evolution of regulatory regions. Dynamics of this kind have features of a random walk on a neutral network, but lead to quantitative changes in the form of the turnover of functional sites over time, while maintaining the function of the regulatory region.

Royal Road Fitness Functions: Some of the simplest fitness functions that demonstrate the properties of neutral subbasins linked by narrow pathways are the RR fitness functions [13-17]. A whole family of RR fitness functions has been proposed, namely R1, R2, R3, and R4 [33]. These functions were specifically proposed for testing the BB hypothesis, and whether recombination actually manipulated such BBs in the way that traditional GA theory assumed [33;34;35;36]. Recently, elaborations such as the RS and Terraced Labyrinth fitness functions were introduced [13-16]. All of these functions demonstrate neutral subbasin architecture. The difficulty of the RR functions increases from R1 to R4.

The function R1 is computed very simply: a bit string x gets 8 points added to its fitness for each of the given order-8 schemas of which it is an instance:

$s1 = 11111111**; c1 = 8$
$s2 = ********11111111**; c2 = 8$
$\cdots\cdots$
$s8 = **11111111; c8 = 8$
$s_{opt} = 1111111111111111111111111111111111111111111111111111111111111111; c_{opt} = 64.$

The value R1(x) is the sum of the coefficients cs corresponding to each given schema of which x is an instance. Here, c_s is equal to order(s). The fitness contribution from an intermediate stepping stone (such as the combination of s1 and s8) is thus a linear combination of the fitness contribution of the lower level components. This fitness function is an example of a class of functions with subbasin-portal architecture (Figure 2). The genotype space consists of all bit-strings of length 64 and contains 9 neutral subbasins of fitness 0, 8, 16, 24, 32, 40, 48, 56 and 64 ([8]). There is only one sequence with fitness 64, 255 strings with fitness 56, 65534 strings with fitness 48, etc.

In the case of the second function R2, the fitness contributions of certain intermediate stepping stones are much higher. R2(x) is computed in the same way as R1, by summing the coefficients c_s corresponding to each of the given schemas of which x is an instance.

The R3 function differs from R2 by the addition of spacers between BBs. In this case, the optimal string has the form of:

$$s_{opt}=11111111********11111111********11111111********11111111********11111111********11111111********11111111********11111111,$$

where an * indicates a random bit and spacer sequences have no impact on the score. The R2 and R3 functions have the same subbasin-portal architecture as R1.

Royal Staircase Fitness Functions: These are a generalization of the RR functions for which the subbasin-portal architecture is more explicit [13-16]. Of all possible configurations of free bits, there is a small subset of portal configurations that lead to increased fitness.

An RS fitness function corresponds to a Terraced Labyrinth whose tree is a simple linear chain. The RS function we use in this paper was defined in a manner similar to RR functions (same length of string, and same size of BB as R1 and R2). Specifically:

$s1 = 11111111**; c1 = 2$
$s2 = 1111111111111111**; c2 = 3$
$\cdots\cdots$
$s7 = 1111111111111111111111111111111111111111111111111111111111111111********; c7 = 8$
$s_{opt} = 1111111111111111111111111111111111111111111111111111111111111111; c_{opt} = 9$

This version of the RS was used in the work of van Nimwegen and Crutchfield [16]. We used it to be able to compare our results to theirs. The genotype space contains 9 neutral subbasins of fitness 1, 2, 3, 4, 5, 6, 7, 8 and 9, and is reminiscent of the R1-R3 function architecture.

One more benchmark test that we are introducing here is the RS3 function (analogous to the RR3). The RS3 function differs from RS by the addition of spacers between BBs. Hence, the optimal RS3 string has the form of:

s_{opt} = 11111111********11111111********11111111********11111111********11111111********11111111********11111111********11111111.

We use this function on some real-life biological tasks below.

2.4. Approach to solve simulations of problems in in vitro evolution

Improvement in the computational tools for in vitro evolution has resulted in new capabilities for modeling molecular evolution. We selected the simulation of forced evolution of transcriptional regulatory regions/modules as an applied case in which the data provides clear constraints. We would like to note, though, that our goal is not to simply produce conclusions on the evolutionary path of certain specific regulatory regions. Rather, our goal is to study the subbasin-portal architecture sequence space for some well-studied regulatory modules and to compare the speed and effectiveness with which our methods solve these problems to other methods used in directed evolution. This provides an excellent test of the approach described in this publication.

Simulation of the Directed (Forced) Evolution of Prokaryotic Promoters: For our target sequence (solution), we selected the sequence of the ribosomal RNA (rRNA) operon promoter rrnP1 in *E. coli*. We chose this particular task because prokaryotic promoters in general and these promoters in particular are without a doubt the most heavily studied [37]. The core promoter of *E. coli* has a length of approximately 60 bp and is characterized by the presence of several conserved sites with spacers in between. It is believed that while the sequence of these spacers is not significant, their length is of extreme importance [9]. There are at least four well-conserved features in a bacterial promoter: the starting point (usually 'CAT'); the -10 sequence ('TATAAT' consensus); the -35 sequence ('TTGACA' consensus); and the distance between the -10 and -35 sequences. We will focus on the strongest type of *E. coli* promoter – the rRNA operon promoter rrnP1. Each rrnP1 promoter sequence contains an AT-rich sequence called the upstream (UP) element [38] upstream of the -35 element. UP elements increase transcription 20- to 50-fold [39]. Its consensus is AAA a/t a/t T a/t TTTT**AAAA, where * indicates a random base, and a/t means A or T. In addition, three to five binding sites for the Fis protein (FisBS) increase transcription three- to eight-fold [39; 40]. (For our tests we used this particular sequence of the FisBS: TGCTGAAAATTTCAGCA) Thus, the target sequence is:

[FisBS]**<~5 bp>**[FisBS]**<~5 bp>**[FisBS]**<~15 bp>**AAA a/$_t$ a/$_t$ T a/$_t$ TTTT**AAAA**<~4 bp>**TTGACA**<16-19 bp>**TATAAT**<5-9 bp>**CAT.

Going by these facts, it is possible to interpret the evolution of the rrnP1 promoter as an example of evolution with the Royal Staircase fitness function. In other words, evolution could proceed by a route starting at a core promoter of reasonable sequence, to a far more powerful promoter with the UP element, to a promoter with maximal strength (another order of magnitude stronger) with a block of FisBSs. As such, this particular version of the Royal Staircase fitness function for the evolutionary search of the rrnP1 promoter has a familiar appearance:

s_1 =
**********…**********TTGACA***…***TATAAT***…***CAT,

s_2 =
**********…**********AAAAATATTTT**AAAA***…***TTGACA***…***TATAAT***…***CAT,

s_3 =
**********…**********TGCTGAAAATTTCAGCA***…***AAAAATATTTT**AAAA***…***TTGACA***…***TATAAT***…***CAT,

.....

s_{opt} =
TGCTGAAAATTTCAGCA…***TGCTGAAAATTTCAGCA***…***TGCTGAAAATTTCAGCA***…***AAAAATATTTT**AAAA***…***TTGACA***…***TATAAT***…***CAT.

For Δ an arbitrary small fitness value, c_1 = Δ, c_2 = ~35 Δ, c_3 = ~100 Δ, c_{opt} = ~150 Δ.

The main difference between the rrnP1 test and the RS3 function is the four-letter alphabet and large words and spacers. Hence it would be crucial for further development of our approach to compare the two tests.

Simulation of the Directed Evolution of Eukaryotic Genes with Multiple Autonomous Regulatory Modules: The group of so called pair-rule genes from the *Drosophila* segmentation network is probably the best candidate to apply our approach for simulations and fitting of models to experimental data. Each such gene is expressed in a series of

seven stripes along the longitudinal axis of the early *Drosophila* embryo (Figure 3). This striped pattern is a blueprint for the process of embryo segmentation which follows. Regulatory regions of these genes consist of a series of autonomous regulatory elements in such a way that each stripe (or pair of stripes in some cases) has its own autonomous regulatory element, or CRM (stripe-specific element). An artificial gene carrying only one such regulatory element, introduced into the *Drosophila* genome, controls expression of its stripe alone. Spacers between the stripe-specific elements (See Figure 3), called insulators, are important also: the CRMs must be separated for proper expression patterning. Hence evolution of these genes is perfectly suited to the Royal Staircase fitness function.

Effectively performing the *in silico* evolutionary search on models of pair-rule genes is a serious challenge. It is a full-scale real-life problem. To attack it we have to use a higher level of abstraction then before. The trick here is to represent the tight clusters of the 4-letter words (specific binding sites for Bicoid, Kruppel, etc., TFs) separated by spacers:

Bicoid&Kruppel Bicoid&Giant Bicoid Hunchback&Giant

...***TTAATCCGTT***...***CGAGATTATTAGTCAATTGC***...***GGATTAGC***...***GAAAGTCATAAAAACACATAATA***...

by a symbol string of a higher level of abstraction:

B K B G G B K B H G B K…***N H H/N N H H N H K H H H***

stripe 2 element stripe 3 element

where B represents a binding site for the Bcd factor, K for Kruppel, H for Hunchback, N for Knirps, G for Giant.

Hence, each symbol represents a binding site (and the spacers inside clusters are ignored). If so, each such cluster (stripe-specific element) is a word or building block at this higher level of abstraction. The BBs are separated by the insulators. Hence, at the higher level of abstraction we again achieved a problem representation similar to the RS3 function.

In our simulations the gene structure ("chromosome" in GA terms) was represented in the form of arrays of numbers. The particular binding site of a given specificity and strength was characterized by a two-digit number: the high-order digit represents the transcription factor to bind (from 0 to 9; e.g. Bcd, etc.), while the low-order digit represents the strength of the site:

10 78 68 48 11 31 66 29 54 29 77 41*87 36 44 73 36 19 38…

stripe 2 element stripe 3 element

I.e., stripe-specific elements are represented as strings of symbols where each two-digit symbol represents one binding site. Each such symbol corresponds to a binding site for factor X (Bicoid, Caudal, Giant, Hunchback, Kruppel, Knirps, Tailless, to be precise) with a strength of Y (discrete values, from 0 to 9).The crucial difference with previous tests is that the fitness landscape is not predetermined, but is calculated via a detailed PDE model (for details see [8]).

A simplified view of the origin of *Drosophila* segmentation assumes that the current way of making segments simultaneously was designed by evolution from ancestral mechanisms for sequential production of segments. This sequential production of stripes was based on temporal oscillations of the ancestral genes in a growing zone. In this picture, the easiest way to imagine how *Drosophila* segmentation arose would be for the evolutionary search to find and substitute ancestral stripe-specific elements, one by one, into the sequential, time-oscillating mechanism (Figure 3). In this scenario, finding the CRM for the 1st stripe would give a score = Δ; finding the CRM for the 2nd stripe would double the score (2Δ); and so on to completion (score=7Δ). (Note that contemporary pair-rule genes have less than 7 stripe-specific elements - the *eve* gene has five - which is why some elements have to control two stripes (for details see [8]).)

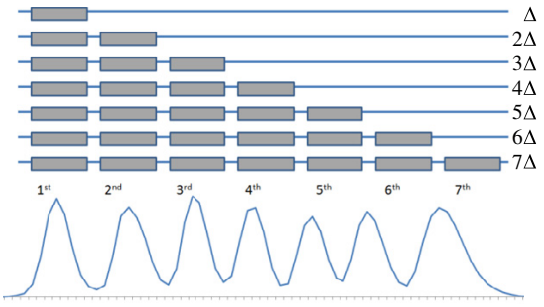

Figure 3 A simplified scenario for the evolutionary origin of pair-rule gene organization: from the ancestral gene, to the case with a single stripe-specific element (top), down to the gene with seven stripe-specific elements. Gene organization and intensity vs. distance for the corresponding patterns of gene expression (bottom) are shown schematically, along with the score in units of Δ.

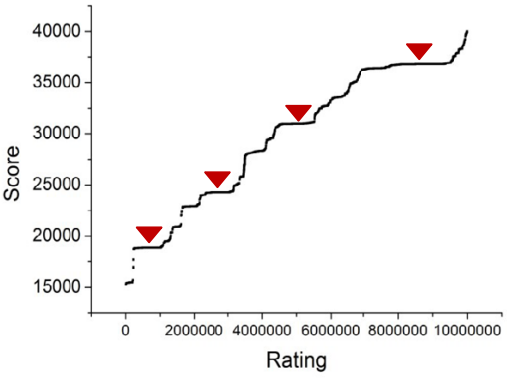

Figure 4 Solution scores in the vicinity of good ones (found by evolutionary search), plotted as a rank distribution, showing the flat sub-basins (red arrow-heads) and bridges (portals) between them.

This simplified view of the evolutionary origin of *Drosophila* segmentation is perfectly fit to the tools for forced *in silico* evolution developed in this publication. The point mutation operator substitutes a symbol, changing the specificity or the strength of the site. Both standard GA and our original crossover operators manipulate the strings as described in previous sections.

3. The Results

As noted above, it is believed that such simple fitness functions as RR and RS reflect to a great degree the significant properties of biological evolutionary searches. These functions are well-studied and are sufficiently simple to permit statistical analysis, and comparison of their theoretical results with the results of experimental runs. Both RR and RS function tests used the same suite of programs [8]. The RC operator takes 15% more computational time than standard GA crossover. The GRC operator is about twice as expensive as standard crossover. The implementation of the RetroGA operators (for binary and symbolic strings) has one main new procedure - the search for areas of local homology. This borrows a common procedure from bioinformatics to apply in GA.

3.1. Royal Road class fitness functions

We used four functions, R2, R3, RS & RS3, that differed in regards to the effectiveness of various evolutionary and non-evolutionary techniques (Table 1 & [8]). Experiments with the RC operator did not show improved efficiency over standard GA for the RR functions, but did show improvement for RS. The GRC operator showed improved efficiency for all functions, and was the most effective of all strategies tested. On every benchmark test (Table 1) it outperformed all others by a significant margin.

With the GRC operator, we found an evolutionary technique that was sufficiently effective on these classical problems to outperform standard GA by a factor of 3 to 4 on all RR functions (Table 1& [8]). To our surprise, the RC operator found answers to the RS tests four times faster on average than standard GA, while the GRC operator did it more than three times faster (Table 1). The presence of spacers in RS3 does not greatly interfere with the efficacy of the RC and GRC operators.

Table 1. Performance of our techniques versus standard GA. Values indicate number of function evaluations needed to reach optimum, averaged over 1000 runs.

Function	*TECHNIQUES*		
	Std. GA	RC operator [8]	GRC operator [8]
R2	73,563±1,794[34;36]	101,986±32,671	15,449±18,163
R3	75,599±2,697[34;36]	136,566±77,241	16,720±29,977
RS	~500,000	122,855±66,459	152,630±129,750
RS3	~500,000	135,141±76,427	163,314±158,295
rrnP1, 4th level	~46,000,000	21,304,252±2,632,548	23,411,266±3,097,116

3.2. Real-life problems

The Bacterial Promoter rrnP1: The previous benchmark tests showed both the RC and GRC operators to be more effective than GA in evolutionary searches for the problems inspired by real eukaryotic molecular genetics (Table 1). But the rrnP1 task is as hard as the R4 test: only a small fraction of runs were able to find the highest-score (5th) level (cf. [8]). In achieving the 4th level, though, the RetroGA operators (RC, GRC) were more effective than standard crossover (GA). Our tests also show that the 4-letter strings and large building blocks make this task substantially more time consuming, compared to the RR/RS tests.

The Enhancer of the Drosophila even-skipped Gene: Our computational experiments with the eve-gene task revealed that its fitness landscape has some key characteristics of the sub-basin & portal architecture. For instance, if we pick a seven-stripe solution (found by evolutionary search) and study the vicinity of this solution in parameter space, we can see that the fitness values belong to flat (iso-fit) areas with short and steep bridges between them (Figure 4). (And note that the fitness landscape was not predetermined by the fitness function definition (as with RR/RS functions), but was calculated via a dynamic PDE model (for details see [8]).)

Table 2 Performance of our technique versus standard GA. Values indicate number of function evaluations needed to reach optimum, averaged over 100 runs.

Function	*% for tests*	*% for control*	*TECHNIQUES*		
			Std. GA	RC operator	GRC operator
eve-gene, 4th level	94	92	3,973,563±919,794	1,656,372±652,671	1,903,877±734,621

The evolutionary search for the *eve*-gene problem turned out to be very time-consuming. However we managed to use the same overall scheme for the search. Namely, each run lasted to a predetermined number of evaluations (5,000,000). Because we did not know the geometry of the fitness landscape in this case, we had to use a threshold criteria to judge whether a given building block (stripe-specific element) was able to control the formation of its stripe (or pair of stripes, see p. 6). The moment the first solution overcame the last threshold (five elements – five thresholds) corresponded to the score for a given run (making these tests comparable quantitatively to the previous ones). To our surprise, most control runs and all test runs achieved the desired seven-stripe pattern, though only 6% of the test runs achieved the last, 5th level. These tests show that the RetroGA operators facilitate the evolutionary search significantly, over GA (Table 2). Interestingly enough, the *eve*-gene task appears substantially less hard than the rrnP1 one (Table 1). We expect this is caused by substantial redundancy of the stripe-specific element structure in the case of the *eve*-gene problem, but it needs further study.

In addition, RetroGA effectively prevents premature convergence of the evolutionary search: the proportion of searches failing to find the desired gene architecture was several times smaller for RetroGA than for standard GA operators. Hence, RetroGA is an advance on the computational tools for *in silico* evolution.

4. Discussion and Conclusions

The mechanisms of diversification in natural GA are not analogous to the mutation and crossover operators in computational GA. In computational GA, these mechanisms are global, act statistically upon the entire population, and use predetermined parameter values. In natural GA, however, the character of the mutation depends on the sequence of the given gene. It may be said that a gene contains not only information that is used to determine its fitness, but also instructions on how to mutate itself afterwards. As such, mutation operators in natural GA are local, and their action depends on the sequence of the particular gene in question.

The positions of crossover sites and exchange between two strings in computational GA are chosen randomly. However, in biology, crossover occurs at sites of high homology between two molecules of nucleic acid. These regions of high homology may be naturally interpreted as BBs. As such, crossover operations in the natural world do not destroy BBs, but instead conserve them wholly, while the material between the BBs undergoes crossover exchanges and point mutations. It is well-known that the destruction of already-discovered BBs by the crossover operator is one of the major problems with GA, and was originally brought to light by experiments with the RR fitness functions. Because of this, the capability of homology-based PCR techniques to conserve already located BBs is of tremendous interest. We think that the improvement of RetroGA over standard GA for fitness functions with sub-basin & portal architecture is caused by the capability of the new operators to preserve BBs already found by the algorithm (via search for local homology).

We believe that methods of discrete optimization developed by the living world have significant meaning for interdisciplinary research. The new algorithms for evolutionary computation that we borrow from the living world are to a significant degree domain-independent. Because of this, they may be easily implemented in various EC techniques.

In the past several decades, computational GA has become an effective mathematical instrument for modeling and analyzing the processes and mechanisms of biological evolution. RetroGA has the potential to have the same effect on molecular evolution. Thus, further study and development of retroGA would serve to lay the foundation of a mathematical theory describing the processes and mechanisms behind the evolution of biological macromolecules.

Acknowledgements

This work is supported by the Joint NSF/NIGMS BioMath Program, 1-R01-GM072022 and the National Institutes of Health, 2R56GM072022-06.

References

1 Francois P. and Hakim V., (2004) *Proc. Natl. Acad. Sci. USA* **101**: 580-5.

2 Paladugu, S., Chickarmane, V., Deckard, A., Frumkin, J., McCormack, M., and Sauro, H. (2006) *IEE Proceedings-Systems Biology* **153**: 223–35.

3 Quayle A. and Bullock S., (2006) *J. Theor. Biol.* **238**: 737-53.

4 Cooper M.B., Loose M., and Brookfield J.F. (2008) *Biosystems* **91**: 231-44.

5 Cooper M.B., Loose M., and Brookfield J.F., (2009) *Biosystems* **96**:185-93.

6 Hardway, H., Mukjopadhyay, B., Burke, T., et al., (2008) *J. Theor. Biol.* **254**: 390-9.

7 Spirov A.V. and Holloway D.M. (2008) In: *Advances in Computational Algorithms and Data Analysis, Lecture Notes in Electrical Engineering* **14**, Springer, pp. 29-50.

8 Spirov A.V. and Holloway D.M. (2009) *Nature Precedings* <http://dx.doi.org/10.1038/npre.2009.3913.1>

9 Lewin B. (2003) *Genes VIII*. 1056 p.

10 Donelson J.E. (1995) *J. Biol. Chem*. **270**:7783-6.

11 Negroni M. and Buc H. (2001) *Annu. Rev. Genet*. **35**: 275-302.

12 Stemmer W.P. (1994) *Proc. Natl. Acad. Sci. USA* **91**:10747-51.

13 Crutchfield, J.P. and van Nimwegen E., (2001) In: *Evolution as Computation, DIMACS workshop*, Springer-Verlag, New York.

14 van Nimwegen E. and Crutchfield J.P. (2000) *Computer Methods in Applied Mechanics and Engineering* **186**: 171-94.

15 van Nimwegen E. and Crutchfield J.P. (2000) *Bulletin of Mathematical Biology* **62**: 799-848.

16 van Nimwegen E. and Crutchfield J.P. (2001) *Machine Learning* **45**: 77-114.

17 van Nimwegen E., Crutchfield J.P. and Mitchell M. (1999) *Theor. Comput. Sci.* **229**: 41-102.

18 Stemmer W.P. (1994) *Nature* **370** (6488): 389-91.

19 Gavrilets S. (2003) In: Crutchfield, J. and P. Schuster (eds.) *Towards a Comprehensive Dynamics of Evolution - Exploring the Interplay of Selection, Neutrality, Accident, and Function,* Oxford University Press, pp. 135-62.

20 Holland J.H. (1992) *Adaptation in natural and Artificial Systems: an introductory analysis with applications to biology, control and artificial intelligence*, MIT Press.

21 Goldberg D.E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, Massachusetts.

22 Wright S. (1982) *Evolution* **36**: 427–43.

23 Kauffman S.A. and Levin S. (1987) *J. Theor. Biol*. **123**:11-45.

24 Kimura M. (1983) *The Neutral Theory of Molecular Evolution*, Cambridge: Cambridge University Press.

25 Fontana W. (2002) *BioEssays* **24**: 1164-77.

26 Held D.M., Greathouse S.T., Agrawal A. and Burke D.H. (2003) *J. Mol. Evol.* **57**: 1-10.

27 Carroll, S.B. (2000) *Cell* **101**: 577-80.

28 Stern D.L. (2000) *Evolution Int. J. Org. Evolution* **54**: 1079-91.

29 Costas, J., Casares, F., and Vieira, J. (2003) *Gene* **310**: 215-20.

30 Ludwig M.Z., Bergman C., Patel N.H., and Kreitman M. (2000) *Nature* **403**: 564-7.

31 Ludwig M.Z. (2002) *Curr. Opin. Genet. Dev*. **12**: 634-9.

32 Kimura, M. (1985) *J. Genet*. **64**: 7-19.

33 Mitchell M. (1996) *An Introduction to Genetic Algorithms*, MIT Press.

34 Mitchell M., Forrest S. and Holland J. H. (1992) In: *Proceedings of the First European Conference on Artificial Life*, Cambridge, MA: MIT Press/Bradford Books.

35 Forrest S. and Mitchell M. (1993) In: D. Whitley (ed.), *Foundations of Genetic Algorithms* **2**, pp. 109-26, San Mateo, CA: Morgan Kaufmann.

36 Mitchell M., Holland J. and Forrest S. (1994) In: J. Cowan, G. Tesauro, and J. Alspector, *Advances in Neural Information Processing Systems*, Morgan Kauffman, San Francisco, CA.

37 Schneider D.A., Ross W., and Gourse R.L. (2003) *Curr. Opin. Microbiol*. **6**:151-6.

38 Ross W., Aiyar S.E., Salomon J., and Gourse R.L. (1998) *J. Bacteriol*. **180**: 5375-83.

39 Hirvonen C.A., Ross W., Wozniak C.E., et al., (2001) *J. Bacteriol*. **183**: 6305-14.

40 Ross W., Thompson J.F., Newlands J.T., and Gourse R.L. (1990) *EMBO J*. **9**: 3733-42.