Biomimetic Representation in Genetic Programming

Michael A. Lones and Andy M. Tyrrell
Bio-Inspired and Bio-Medical Engineering
Department of Electronics
University of York
Heslington, York, England
{Michael.Lones, Andy.Tyrrell}@bioinspired.com

Abstract

Biological representations underly biological evolution. Moreover, they are also a product of evolution and consequently well adapted for their purpose. The argument presented in this paper is that the representations of biology are also suitable for representing artificial executable systems in genetic programming and, furthermore, that biomimetic representations could improve both the adaptability and evolvability of GP. To this end a biomimetic approach to GP, enzyme genetic programming, is introduced and its behaviour is analysed when applied to the domain of combinational circuit design.

1 INTRODUCTION

Whilst it is true that manual programming and genetic programming (GP) explore the same search space, the manner in which they do so is very different. Manual programming is a design process. Moreover, it is a design process necessarily biased by the cognitive limitations of human programmers. Consequently, the tools, languages and artifacts of manual programming make implicit the linearity and decompositional constraints of human thought. Evolution does not require limitations since it does not require understanding of the domain. Indeed, experience in evolutionary computation (EC) suggests that placing unsuitable constraints upon a search space can reduce performance of evolutionary algorithms (EAs) and even restrict search from traversing sub-spaces containing global optima. Hence there is a danger that the use of manual programming representations in GP could place unnecessary constraints upon the scope and evolvability of evolutionary search.

Rather than design a new representation from scratch, the approach taken here is to mimic biological representations. The reason for this is two-fold. One, biology has already proven a useful source of ideas in EAs, and in computer science more generally. Two, biology and GP share important characteristics. The obvious similarity is that both apply evolution to their respective products. However, a less obvious similarity between GP and biology is that both their products are executable; in that static descriptions are mapped to dynamic systems — systems that, furthermore, can be expressed using a common representation [2]. From this it follows that GP, a system sometimes accused of lacking evolvability, could benefit through greater concordance with biology, a system with proven evolvability at a broadly similar task.

The paper proceeds as follows. Section 2 describes biological representations. Section 3.1 discusses the use of biomimetic representations in GP and artificial computation. Section 3.2 describes enzyme genetic programming, a biomimetic GP system.

2 THE BIOLOGICAL APPROACH

Development in biology is a mapping from a genotype, a static description of an organism, to a phenotype, a dynamic instance of the organism. This mechanism can be described at a number of different levels, the most fundamental of which is the level of gene expression; where genomic information is used to construct proteins, the basic elements of biological phenotypes.

2.1 GENE EXPRESSION

Gene expression, the process of transforming a gene into a protein, is often expressed as a combination of transcription; copying the gene, and translation; converting the language of the gene, DNA bases, into the language of the protein, amino acids. More accu-

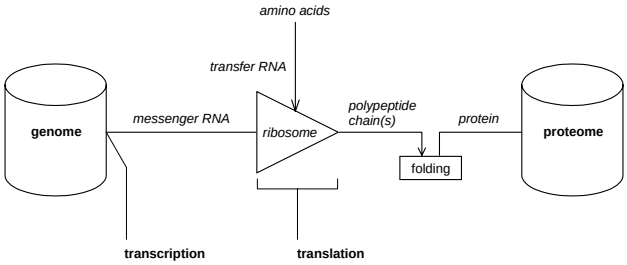

Figure 1. Gene expression

rately, transcription and translation are components of a transformation process which, via the orchestration of many biological agents (that were also at an earlier time generated by this process), builds proteins according to specifications recorded in an organism's DNA. A computational interpretation of this process is depicted in figure 1. The genome is envisaged as a database of descriptions of computational agents, stored in some random access storage media. Transcribers (protein complexes organised around DNA polymerase) extract records from the genomic database and record them in temporary linear data structures (messenger RNA) which, after editing to remove obsolete sections, are delivered to translators. Translators (complexes of protein and RNA called ribosomes) parse this data, using it as a blueprint to guide construction of the components required for new computational agents. However, before they are assembled, these components, linearly-connected sequences (polypeptides) of basic functional units (amino acids), are folded into three-dimensional shapes; where the spatial arrangements of their functional units determine the computational natures of the components. New computational agents (proteins) are then formed by the physical composition of one or more of these components. The set of all proteins generated by way of gene expression within an organism is called the proteome.

2.2 BIOCHEMICAL PATHWAYS

If a cell is viewed as a computational system, then that computation takes the form of the manipulation, the metabolism, of the cell's chemical state; the concentrations of different chemical species within the cell. Manipulation of this state involves transformations, implemented by constructive and destructive chemical reactions, between chemical species; increasing the concentrations of some and reducing the concentrations of others. However, temperatures in biological systems are relatively low and many of these reactions will not occur without the help of catalytic agents — enzymes, a group of proteins that bind to specific chemicals and mediate their interaction and subsequent transformation. The chemical species to which an enzyme binds, its substrates, is determined by a property called its specificity; a result of the spatial arrangement of amino acids found at the enzyme's binding sites.

The presence of enzymes activates transformative paths within the metabolism. This forms a network, a metabolic network, where chemical species are nodes and enzyme-mediated reactions are vertices. Metabolic networks are composed of metabolic pathways. A metabolic pathway is an assemblage of enzymes which, roughly speaking, carries out a unified task involving a shared group of chemical species. This cooperation emerges from the sharing of substrates between enzymes, where the product of one enzyme becomes the substrate of another. Pathways can be linear, forked or iterative; iteration resulting where a product feeds back to an earlier stage in the pathway. However, the metabolism operates within a dynamic environment and therefore requires a dynamic execution structure to cope with varying demands. Executional dynamism is controlled either directly, through regulation of pathways, or indirectly, through expression of enzymes. Pathway regulation acts through the regulation of enzymes; either through chemical modification or by effector molecules that bind to enzymes and alter their activity. The source of regulation may be this pathway, another metabolic pathway, or a signalling pathway; a type of biochemical pathway which delivers external signals to cells.

Enzymes are proteins and are therefore a result of gene expression. Gene expression offers the potential for the synthesis of all proteins found within the proteome, yet due to the presence of regulatory mechanisms, each cell generates only a fraction of this potential. Genes are predominantly regulated by the interference of DNA-binding proteins upon the formation of their transcription complexes, interference that can be negative or positive. Since gene expression is regulated by products of gene expression, genomes are ultimately self-regulating. The regulatory interactions between genes and their products are described by a third class of biochemical pathway; gene expression pathways.

2.3 EVOLUTION

From a human perspective, the genome seems static; a single database that is faithfully copied and disseminated throughout the lifetime of an organism. From an evolutionary perspective, the genome is dynamic; growing, mutating, recombining, rearranging — evolving between the lifetimes of single organisms.

If evolution is considered as a search process, then one interpretation of genomes is that they represent current search points within a landscape of possible genomes. However, a different interpretation is that genomes represent the state of search within evolution; they contain not just information relevant to the present in search but also control information — information describing the history and potential future of search. Historical information is important for backtracking, and hence breadth of search, whereas information regarding the future conceivably improves depth of search.

Diversity amongst genomes is sustained by the presence of multiple alternative alleles at a single gene locus within a population. According to the neutral theory of evolution, this phenomenon, called polymorphism, represents transitory events where alternative alleles at a gene locus are heading towards either fixation or extinction. Neutralism, backed up by measurements of the degree of polymorphism within natural populations, proclaims that evolution occurs mostly through the random fixation of neutral mutations and the removal of degenerative alleles. Advantageous mutations, by comparison, are rare.

Evolution is the population-wide genetic change of a species. Adaptation is the improvement of a species' fitness. Whilst neutral mutation drives evolution, adaptation is the result of positive mutations. However, positive mutations are only likely to occur where a current allele is close to a better allele in genotypic space. The action of neutral evolution is to explore different areas of a region of fitness equivalence within genotypic space. A neutral walk is a series of such mutations. If the walk takes the allele close to a region of higher-fitness space, then a positive mutation is more likely to occur. Hence, neutrality increases exploration and diversity, improving the likelihood of an advantageous mutation occurring.

3 GENETIC PROGRAMMING

3.1 BIOMIMETIC APPROACHES

Biological representations possess a number of properties which improve their adaptability and evolvability. For genetic representations, these include neutrality, multiplicity of expression and positional independence. For phenotypic representations, they include expressiveness, reconfigurability and fault tolerance. Adaptability and evolvability are, of course, also goals of GP. The following sections review previous approaches which have used mimicry of biological representations in an attempt to achieve these goals.

3.1.1 Developmental representation

The most common form of biomimicry found in the representations of GP systems is the genotype-phenotype mapping, the separation of variational and generative solution representations by a developmental stage. The benefit of this separation is that it allows the genetic representation to be improved with respect to evolution without affecting the representation used for the executable structure. An example of this in GP is grammatical evolution [3]. Programs in grammatical evolution are defined by an arbitrary BNF grammar. Genotypes are expressed as sequences of production choices within the BNF grammar. During development, this sequence, the blueprint of the program, is mapped to an executable program. The representation used for production choices is degenerate, meaning that a single choice can be encoded in multiple distinct ways; leading to support for neutral mutation and neutral walks. A more general benefit of a redundant mapping from genotype to phenotype, suggested in [4], is the resulting increase in genetic diversity.

3.1.2 Genetic Representation

In grammatical evolution, genes are production choices. However, the relationship between gene products is based upon genomic ordering. One approach which limits positional dependency in GP is gene expression programming [5] which segments a program into small parse trees which are then composed by a global function. Angeline's MIPS nets [6] also express programs as collections of interacting parse trees, though not in an intentionally biomimetic sense. Another, overtly biomimetic approach is described in [7], in which finite state automata are expressed as a multiset of genes; each of which consists of a state reference and a pattern. This pattern is then used as a template for deciding which other states the gene will interact with; and hence which state transitions will be formed.

3.1.3 Phenotypic Representation

The use of gene expression, the mimicry of biological genetic representations, is a developing notion in GP [8]. Less developed is the abstraction of biological phenotypic representations; which can be regarded as the natural mapping for systems described genetically. The idea of abstracting biological computational systems for artificial use is as old as computers themselves; and artifacts like neural networks and evolutionary algorithms show it to be an effective strategy. Furthermore, the basis of phenotypic representation — proteins — and higher-level metabolic and signalling

systems have been shown to be computational in nature. Fisher [9] describes the computational abilities of enzymes, and Bray [10] the computational behaviours and simulated evolution of protein networks. Further evidence is given by the modelling of metabolic networks as Petri nets [2]. An information-processing artificial enzyme system is demonstrated in [11].

3.2 ENZYME GENETIC PROGRAMMING

Enzyme genetic programming [1] is a GP system that mimics both genetic and phenotypic representations to evolve analogues of metabolic pathways in artificial programmable systems. The model of a metabolic pathway used in enzyme GP is a set of interacting functional elements where the nature of interaction is decided locally by a unit according to its specificities — defined in the genome — for outputs from other units. An important difference between this and other GP systems that describe programs as networks (for example, [12]) is that here functional elements can have specificities for any number of other components, not just those they ultimately interact with.

Enzyme GP has so far been applied to problems in combinational logic design; the evolution of non-recurrent digital circuits. Other research in this area is presented in [13]. The mapping between metabolic networks and digital circuits is fairly natural. Logic gates, like enzymes, enable transitions to occur in the space of possible states, and circuits, like metabolic pathways, enable functional sequences of transitions; which in this case carry out a combinational logic function upon the circuit inputs.

3.2.1 Overview

Circuits, as illustrated in figure 2, have both genetic and phenotypic forms. The genetic form is a linear array of genes, where each gene consists of an activity and a list of specificities for the outputs of other activities. Activities are input terminals, output terminals or instances of boolean functions. The activities available to evolving circuits are recorded in a global 'gene pool' which is defined at the beginning of an evolutionary run. Specificities are numerical values in the range 0 to 1.

Once expressed, gene products behave in an enzymatic fashion, sinking products from those activities they have the highest specificity for and transforming them into other products depending upon their activity. However, the circuits generated by the method must be viable and therefore non-recurrent. Rather than constrain the genetic representation or discard

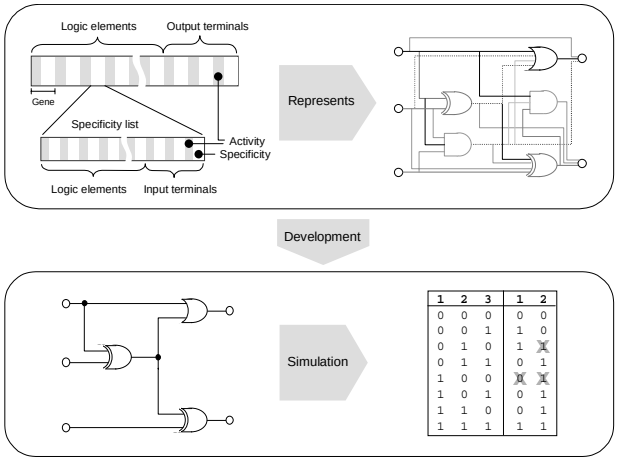

Figure 2. Circuit representations

invalid descriptions, all genotypes are mapped to valid circuits; meaning that activities must choose inputs for which they have 'recessive' specificity if their most favoured inputs already occur downstream in the developing circuit.

Evolution of circuit genotypes is enacted with a distributed diffusion-model genetic algorithm. This algorithm, detailed in [1], applies an elitist evolution strategy at each node of a spatially-distributed population. Variation is applied through uniform crossover, between the specificities of homologous activities, and mutation, randomly modifying a proportion of a solution's specificities.

3.2.2 Evaluation

Enzyme GP's performance has been measured upon two problems; the full adder and the two-bit multiplier. The algorithm is given a truth table and the gates found in the standard circuit. Its aim is then to form the correct connections between the circuit elements. Results for the adder are graphed in figure 3. Results for the multiplier are recorded in table 1. Cumulative mean gives the average number of generations between optimal solutions for a series of runs. Computational effort is a standard measure [14] which allows a degree of comparison between EAs. For the adder, the minimum effort is about 42000 evaluations for a 99% certainty of success. Values for the multiplier, shown in the table, can be compared to results from [13], where minimum effort is cited as between 210000 and 585000 depending on the choice of available logic functions. Each system places different constraints on the search space so direct comparison of performance is not possible. Nevertheless, the performance of enzyme GP seems no worse than Miller's system.

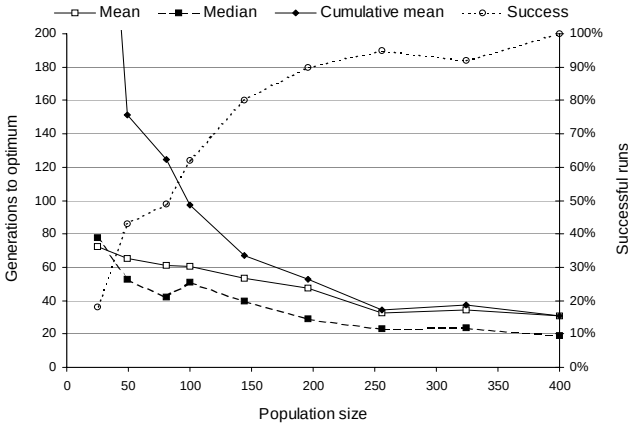

Figure 3. Results for full adder

Population size	Generations		Successful runs	Average suboptimum	Computational effort
	Mean	C.Mean			
196	69	196	35%	95.6%	213248
324	70	127	55%	96.4%	252720
400	56	93	60%	96.2%	179200

Table 1. Results for 2-bit multiplier

3.2.3 Analysis

Figure 4 shows a typical evolution of a full adder; showing how the evolutionary process transforms twelve first generation random solutions into a thirtieth generation correct solution. The expanded section shows the nature of phenotypic variation caused by genetic operators. It should be apparent that simple genotypic variation is not reflected as simple phenotypic variation. For instance, circuit D is a simple mutation of circuit A's specificities, yet the effect is more akin to a compound macromutation; with the introduction of a new circuit element in tandem with the modification of another element's input. For most mutations, there is no phenotypic change, suggesting that the major role of mutation is the generation of neutral diversity through synonymous genetic changes. A series of such changes is a neutral walk. Returning to figure 4, the transition between the final crossover and the optimal solution is a neutral walk which brings the genotype near enough to the optimum for it to be found by a single further mutation.

The action of crossover upon genotypes also leads to non-trivial effects upon the phenotype. Recombining circuits D and E, for example, leads to circuit F; where the output terminal receives input from a gate to which it was connected in neither parent. The unexpected phenotypic behaviours of crossover are due to changes in the dominance hierarchies of the genotype; resulting in recessive, yet relatively strong, specificities becoming expressed when dominant specificities are removed. This also reflects the multiplicity of solutions in enzyme GP. In addition to the dominant schema which are expressed in any given generation, a genotype also carries many recessive schema. Changes in the dominance hierarchy, caused by the action of genetic operators between generations, means that a large number of different, yet related, solutions can be expressed within different generations.

Multiplicity, a single genotype describing many related phenotypes, is conceivably more efficient than having many genotypes each mapping to a single phenotype since (i) it reduces computational effort within a generation and (ii) it keeps related information in one place rather than distributing it throughout the population. It also reflects the view that genomes are adaptive because they elicit state of search in addition to position in search. In this case, the genotypes contain components of previous and future solutions in addition to those expressed in the current solution; and consequently possess more information regarding the direction of search.

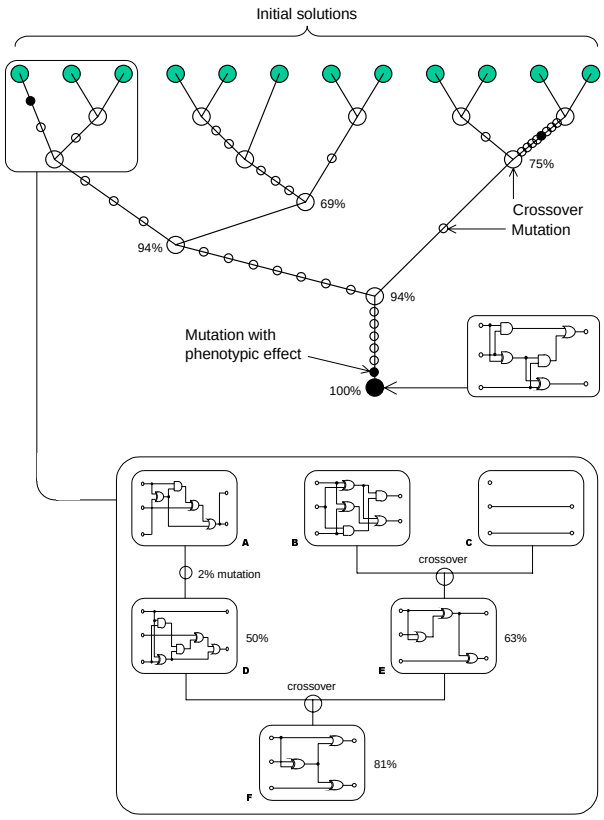

Figure 4. Evolution of a full adder

However, it seems likely that enzyme GP suffers from too much multiplicity. In each solution, every circuit element that requires inputs has specificity for every element that provides output. This implies a large ratio of recessive components to dominant components; and since fitness is measured only on dominant components, this may encourage the hitchhiking of low-fitness recessive components. The disparity between solution fitness and true evolutionary worth may in turn limit the effectiveness of selection.

4 CONCLUSIONS

The argument of this paper is that representations in GP could be improved through mimicry of biological representations. Enzyme genetic programming, whilst not yet proving this assertion, shows that adaptation of biological representations for use in the artificial domain is at least possible. Early results, applying the method to combinational logic design, show the method to be competitive, whilst analysis of its behaviour suggests that biomimetic representations may be advantageous to artificial evolution.

5 FUTURE WORK

The behaviours of the three classes of biochemical pathway; metabolic, signalling and gene expression, have been described, respectively, as self-organising, self-reshaping and self-modifying [15]. Much of the power of biology comes from the reshaping and modifying activities of the latter two classes of pathway. Consequently, an interesting direction of research would be the addition of these other axes of behaviour into the enzyme GP system.

References

[1] M. A. Lones and A. M. Tyrrell. Enzyme genetic programming. Accepted for CEC2001, May 2001.

[2] V. Reddy, M. Mavrovouniotis, and M. Liebman. Petri net representations in metabolic pathways. In L. Hunter et al, editor, *Proceedings of the first international conference on intelligent systems for molecular biology*. MIT Press, 1993.

[3] C. Ryan, J. J. Collins, and M. O'Neill. Grammatical evolution: Evolving programs for an arbitrary language. In W. Banzhaf et al, editor, *First European Workshop on Genetic Programming*, volume 1391 of *Lecture Notes in Computer Science*. Springer, April 1998.

[4] R. E. Keller and W. Banzhaf. Genetic programming using genotype-phenotype mapping from linear genomes into linear phenotypes. In J. Koza et al, editor, *Genetic Programming 1996: Proceedings of the First Annual Conference*. MIT Press, 1996.

[5] C. Ferreira. Gene expression programming: A new adaptive algorithm for solving problems. Unpublished, manuscript available at www.gene-expression-programming.com, 2000.

[6] P. Angeline. Multiple interacting programs: A representation for evolving complex behaviors. *Cybernetics and Systems*, 29(8):779–806, 1998.

[7] S. Luke, S. Hamahashi, and H. Kitano. "Genetic" Programming. In W. Banzhaf et al, editor, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, 1999.

[8] H. Kargupta. Gene expression: The missing link in evolutionary computation. In D. Quagliarella, J. Periaux, C. Poloni, and G. Winter, editors, *Genetic Algorithms in Engineering and Computer Science*, chapter 4. John Wiley & Sons Ltd, 1997.

[9] M. J. Fisher, R. C. Paton, and K. Matsuno. Intracellular signalling proteins as 'smart' agents in parallel distributed processes. *BioSystems*, 50:159–171, 1999.

[10] D. Bray. Protein molecules as computational elements in living cells. *Nature*, 376:307–312, 1995.

[11] M. Shackleton and C. Winter. A computational architecture based on cellular processing. In *International conference on Information Processing in Cells and Tissues (IPCAT)*, 1997.

[12] J. Miller and P. Thomson. Cartesian genetic programming. In R. Poli et al, editor, *Third European Conference on Genetic Programming*, volume 1802 of *Lecture Notes in Computer Science*. Springer, 2000.

[13] J. F. Miller, D. Job, and V. K. Vassilev. Principles in the evolutionary design of digital circuits — part I. *Genetic Programming and Evolvable Machines*, 1:7–36, April 2000.

[14] John Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, 1992.

[15] P. C. Marijuán. Enzymes, artificial cells and the nature of biological information. *BioSystems*, 35:167–170, 1995.