ABSTRACT

COMPUTER EVOLUTION OF GENE CIRCUITS FOR CELL-EMBEDDED COMPUTATION, BIOTECHNOLOGY AND AS A MODEL FOR EVOLUTIONARY COMPUTATION

by

Tommaso F. Bersano-Begey

Chair: John H. Holland

This dissertation describes how to improve automated design and evolution in computers using the structuring of genetic programs in biological systems. It also shows how to 'reprogram' cells to perform useful tasks by embedding computer-evolved code in biological organisms. This reprogramming makes it possible to exploit in new contexts the cell's ability to self-repair, replicate, and generate chemicals, light, or motion at microscopic scales.

Biological systems evolve to create clever designs for solving problems, using mechanisms that do not explicitly involve knowledge or intelligence. The designs that result are often superior to the best human designs. The mechanisms involved reuse and recombine previously discovered structures -building blocks- to generate more complex designs. Such processes reduce problems that grow exponentially in difficulty with size to simpler problems with a hierarchical structure.

AI search techniques like Genetic Algorithms attempt to emulate this mechanism and harness it for automated programming and problem solving and have been successfully applied in a vast number of areas as a powerful black-box design and optimization tool.

However, current implementations still require significant human input in the initial problem formulation (whereas real evolution does not), which also results in different customizations for each problem, making it difficult to determine what characteristics are most useful for their performance. In contrast, biological evolution uses the same structures and mechanisms (DNA) to solve problems as different as flying or optimizing metabolic reactions.

I address these problems by identifying and testing key mechanisms and features responsible for the success of evolution from a computational perspective, and use them to explain previous results and build a single all-purpose implementation that, much like electronic circuits, avoids unwanted human overhead and customization by using the same sub-symbolic building blocks (gates and circuit patterns like loops, counters) for each problem.

I demonstrate its biological soundness by comparing simulation results to human-designed gene circuits and by developing a step-by-step mapping and fabrication of corresponding DNA sequences for a few examples (using genetic engineering to manipulate cellular aging, chemotaxis) which I then insert and test in cells.

© Tom Bersano
All Rights Reserved 2003

TABLE OF CONTENTS

ACKNOWLEDGMENTS ii

LIST OF FIGURES vi

LIST OF TABLES x

PART I Introduction 1

CHAPTER 1 Overview: 4

PART II Foundations 11

CHAPTER 2 Genetic Engineering and Artificial Life. 13

2.1 Artificial Life Research 15
2.2 Biological computation 18
2.3 Problems with human design of Gene Circuits and the need for moving toward automatic computer design 20
2.4 Computer Design and Evolution of Blueprints for Gene Circuits and Genetic Modifications 26

CHAPTER 3 Artificial Intelligence and Biologically Inspired Computer Algorithms 28

3.1 Levels of Organization in Science, Engineering and Artificial Intelligence: 30
3.2 Research Directions in Artificial Intelligence 30
3.3 Cognitive and Logic-Based Systems 31
3.4 Search Algorithms and Combinatorial Explosion 33

CHAPTER 4 Building Blocks in Problem Solving and Design 36

4.1 The Importance of Representation and Encoding in Evolving Systems 38
4.2 Choosing the proper level of representation 41
4.3 Finding the Elusive Building Blocks in Computer Evolution 43

PART III Addressing unresolved issues in Computer Evolution Theory 48

CHAPTER 5 Revising Computer Evolution through Molecular Genetics and Sub-symbolic Building Blocks 51

5.1 The Larger Role of Molecular Genetics in Evolution 51
5.2 Descriptive vs. Algorithmic Representation 52
5.3 A common problem with previous work reintroducing molecular concepts in GA 57

5.4 Bit rearrangement and position-independent representations 58
5.5 Symbolic vs. sub-symbolic representation and gene expression. 62

PART IV Modeling Biological Programs, evolution, and reuse mechanisms IN computers 65

CHAPTER 6 Revisiting GA and Learning Classifier Systems 67

6.1 Escaping GA's No Free Lunch Theorems 71

CHAPTER 7 The living cell as an operating system for evolution and integration of biological nanomachines 74

7.1 Two main types of cells: Eukaryotes and Prokaryotes 77
7.2 Designing devices from assembly of not fully understood biological nanomachines. 80
7.3 Biological Hardware vs. Abstract Problem Solving 81

CHAPTER 8 Modifications in the Three Aspects of Biological Programs: Execution, Encoding, and Evolutionary Operators: 84

8.1 Execution 85
8.1.1 Protein-protein interactions 85
8.1.2 Fast Reversible and Enzymatic Interactions 86
8.1.3 Signal Degradation and Accumulation 88
8.1.4 Multiple Message lists and compartments 90
8.1.5 Metabolic and Environmental Simulations 92
8.2 Encoding 97
8.2.1 Markers and Floating Redundant Representation 98
8.2.2 Modifiers 101
8.2.3 Multiple regulatory elements and protein domains 102
8.3 Evolution and Genetic Operators 104
8.3.1 Homologous Recombination 104
8.3.2 Transpositions, insertions, deletions and duplications 105
8.3.3 Mutations 106

CHAPTER 9 Summary: Properties of the Ideal Computer Evolution Algorithm 107

9.1 Sub-symbolic building blocks 108
9.2 Supporting and Assembling Mechanisms in Execution: 109
9.2.1 Voting mechanisms and Parallel Execution 110
9.2.2 Variable scoping 110
9.2.3 Developmental assembly of faster decision systems, self-assembly and fractal encoding 111
9.3 Supporting and Assembling Mechanisms in Evolution and design 112
9.3.1 Exploitation vs. exploration by replicate and diverge mechanisms 112
9.3.2 Resistance to damage 113
9.3.3 Fine-tuning and gradation 113
9.3.4 Program reuse by call by reference mechanisms 114
9.3.5 Preserving potentially useful genetic material 114

PART V Experimental Simulations and Comparisons of Search Performance 117

CHAPTER 10 Test problem: repression-only gene circuits 120

10.1 Shorthand notation and encoding...121
10.2 Experiment Set 1: Impact of position independence on recombination and crossover effectiveness...124
10.3 Experiment Set 2: Recombination of provided building blocks in variable-length list representation...127
10.3.1 Negative Effect of List Duplicate node removal 130
10.4 Experiment Set 3: Parallel Graph vs. Sequential Tree Structure 131
10.5 Experiment Set 4: Evolution vs. Standard Search (Hill Climbers)............134
10.6 Simple thought experiment on the advantages of fractal representation 138

PART VI Converting Computer representation of circuits into a Practical Genetic Engineering Implementation 141

CHAPTER 11 Controlling cell-immortalization and aging.....................................143

CHAPTER 12 Simple computer encoded programs for controlling cell-immortalization and aging 152

CHAPTER 13 Genetic Engineering implementation and construction steps 156

13.1 Retrieving full-length gene sequences. 156
13.2 Concatenating DNA sequences...158
13.3 Extracting and copying DNA sequences from organisms and vectors.......160
13.4 Verifying constructs by DNA sequencing 165

CHAPTER 14 Experimental results in living cells..167

PART VII Environment Simulations of Re-programmed Swimming Bacteria 173

CHAPTER 15 Bacterial Chemotactic Swimming: Environment simulation of complex behavior and time dynamics from simple circuits 175

15.1 Experimental Setup 176
15.2 Environment Simulation 177
15.3 Simplified Gene Circuit Representation 178

PART VIII CONCLUSIONS AND FUTURE DIRECTIONS 186

CHAPTER 16 Concluding Remarks...187

16.1 Contributions...187

CHAPTER 17 Future Directions - Toward Larger and More Complex Programs and Implementations..190

BIBLIOGRAPHY 194