EVOLUTIONARY ALGORITHMS AND EMERGENT INTELLIGENCE

DISSERTATION

Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the Graduate
School of The Ohio State University

By

Peter John Angeline, B.S., M.S.

* * * * *

The Ohio State University
1993

Dissertation Committee:

Jordan B. Pollack (Adviser)
B. Chandrasekaran
Feng Zhao

Approved by

_____

Adviser
Department of Computer and
Information Science

© Copyright by

Peter John Angeline

1993

Early zoologists classified as mammals those that suckle their young and as reptiles those that lay eggs. Then a duck-billed platypus was discovered in Australia laying eggs like a perfect reptile and then, when they hatched, suckling the infant platypi like a perfect mammal.

The discovery was quite a sensation. What an enigma! it was exclaimed. What a mystery! What a marvel of nature! When the first stuffed specimens reached England from Australia around the end of the eighteenth century they were thought to be fakes made by sticking together bits of different animals. Even today you still see occasional articles in nature magazines asking, “Why does this paradox of nature exist?”

The answer is: it doesn’t. The platypus isn’t doing anything paradoxical at all. It isn’t having any problems. Platypi have been laying eggs and suckling their young for millions of years before there were any zoologists to come along and declare it illegal. The real mystery, the real enigma, is how mature, objective, trained scientific observers can blame their own goof on a poor innocent platypus.

Zoologists, to cover up their problem, had to invent a patch. They created a new order, monotremata, that includes the platypus, spiny anteater, and that’s it. This is like a nation consisting of two people.

- Pirsig (1992)

For my loving spouse Lorie,

who believed in me even when I had doubts.

Sweetheart, this is all your fault.

8-)

ACKNOWLEDGMENTS

First and foremost I would like to thank my adviser, Professor Jordan B. Pollack, who gave me the intellectual latitude I needed to perform this work and provided such a stimulating yet safe haven for my research. His creativity and curiosity was a continual source of inspiration. I would also like to thank my reading committee, Professor B. Chandrasekaran and Professor Feng Zhao, for their patience while this document came together. I also wish to thank the Office of Naval Research for providing support for this work under contracts #N00014-89-J-1200 and #N00014-92-J-1195.

I wish to give special thanks to Greg Saunders, my collaborator, proof reader, English teacher and friend. Our mutual competitiveness fostered many fruitful discussions and some exhausting racquetball games. The work in Chapter 4 is taken from one of our joint efforts and I gratefully acknowledge his significant contributions to this chapter and many other aspects of this work. I would also like to give special thanks to Dr. David Fogel who befriended me just as I began exploring evolutionary algorithms and encouraged me to explore them fully. My appreciation of his writing, editing and proofreading skills are only surpassed by my admiration of his research ability and clarity of thought.

I would also like to thank the following who helped me in various ways during my life as a graduate student: Ed Large; John Kolen; Mike Weintraub; Richard Fox; Kim Kinnear; John Josephson; Arun Welch; The Laboratory for Artificial Intelligence Research (LAIR); The Ohio State University Center for Cognitive Science (CCS); Graduate Students for Cognitive Science (GradCog); all of the wonderful folks who run the Department of Computer and Information Sciences; all of the Post-symbolic Pre-docs, past and present; and The Dead Researcher's Society (DRS). Thanks also to all of my in-laws for support, Patricia L. Angeline (my mom) for occasional pep talks, and my daughter, Rebecca Lee Angeline, for providing such a wonderful distraction during the preparation of this document.

Finally, I would like to thank Professor L. David Pye whose dedication to his chosen field, genuine love of science and incisive advice was a welcome anchor throughout my graduate career. I am honored to call him my father-in-law and my friend.

VITA

August 14, 1962	Born - Latrobe, Pennsylvania
1984	B.S. Carnegie-Mellon University, Pittsburgh, Pennsylvania
1984 - 1987	Associate Programmer, Control Data Corporation, Bloomington, Minnesota
1987	Advanced Software Engineer, Minnesota Mining and Manufacturing (3M), St. Paul, Minnesota
1988	Artificial Intelligence Consultant
1988 - Present	Graduate Research Assistant, The Ohio State University, Columbus Ohio
1989	M.S. The Ohio State University, Columbus, Ohio

PUBLICATIONS

Angeline, P. (1992) DynaMind Developer: An Alchemist's Primer. *IEEE Expert*, October, 1992.

Angeline, P. (1994) Genetic Programming and Emergent Intelligence. In *Advances in Genetic Programming*, K. Kinnear (ed.), Cambridge MA:MIT Press.

Angeline, P. (1994) A Review of *Genetic Programming*: *On the Programming of Computers by Means of Natural Selection*. *Biosystems*.

Angeline, P. and D'Onofrio, T. (1987) "A Model-based Expert System for Component Level Fault Diagnosis," *Applications of Artificial Intelligence V*, The Society for Photo-Optical Instrumentation Engineers, Orlando, Florida.

Angeline, P. and Pollack, J. (1992) The Evolutionary Induction of Modular Computer Programs. *The Proceedings of The Fourteenth Annual Conference of the Cognitive Science Society*, Bloomington, Indiana.

Angeline, P. and Pollack, J. (1993) Evolutionary Module Acquisition. *The Second Annual Conference on Evolutionary Programming*, La Jolla, California.

Angeline, P. and Pollack, J. (1993) Competitive Environments Evolve Better Solutions for Complex Tasks. *Genetic Algorithms: Proceedings of the Fifth International Conference (GA93),* S. Forrest (ed.).

Angeline, P. and Pollack, J. (1993) Coevolving High-Level Representations. *Artificial Life III*, C. Langton (ed.).

Angeline, P., Saunders, G. and Pollack, J. (1994) An Evolutionary Algorithm that Constructs Recurrent Networks. *IEEE Transactions on Neural Networks*.

Angeline, P., Saunders, G. and Pollack, J. (1994) Evolving Recurrent Neural Networks. In *The Third Annual Conference on Evolutionary Programming (EP94)*.

Saunders, G., Angeline, P. and Pollack, J. (1994) Structural and Behavioral Evolution of Recurrent Networks. In *Advances in Neural Information Processing VI*.

Saunders, G., Kolen, J., Angeline, P. and Pollack, J. (1992) Additive Modular Learning in Preemptrons. In *The Proceedings of The Fourteenth Annual Conference of the Cognitive Science Society*, Bloomington, Indiana.

FIELDS OF STUDY

Major Field:	Computer and Information Science, Professor Jordan B. Pollack
	Specialization: Artificial Intelligence
Minor Field:	Parallel Processing, Professor P. Sadayappan
Minor Field:	Cognitive Science, Professor Caroline Palmer

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGMENTS iii

VITA iv

LIST OF TABLES x

LIST OF FIGURES xi

LIST OF SYMBOLS xvi

CHAPTER **PAGE**

I INTRODUCTION: SEARCH AND EXPLICIT KNOWLEDGE 1

1.1 Literal Knowledge Level Models 2
1.2 The Problems of Explicit Knowledge 4
1.2.1 Credit Assignment Problem 5
1.2.2 The Knowledge Acquisition Bottleneck 5
1.2.3 Memory (Knowledge) Indexing Problem 6
1.2.4 The Problem of Scaling 6
1.2.5 Representation Design 7
1.3 The Root of the Problems 7
1.4 Evolutionary Algorithms 9
1.5 Emergent Intelligence 10
1.6 Demonstrations of Emergent Intelligence 11
1.7 Scope of this Dissertation 12
1.8 Contributions of this Work 13
1.9 A Guide to Reading this Dissertation 14

II BACKGROUND 15

2.1 Search and Intelligence 15
2.2 Optimization and Satisficing 16
2.3 The Credit Assignment Problem 17
2.4 Weak and Strong Methods 17

2.4.1 Weak Methods 18
2.4.1.1 Generate and Test 19
2.4.1.2 Hill Climbing 20
2.4.1.3 Depth First Search 21
2.4.1.4 Breadth-First Search and Beam Search 22
2.4.2 Principles of Standard Weak Methods 23
2.4.3 Representation of Task Environment in Strong Methods 24
2.5 Evolutionary Algorithms 25
2.5.1 Evolutionary Models 27
2.5.2 The Selection of Surviving Objects 28
2.5.3 Variations of the Evolutionary Algorithms 30
2.5.3.1 Genetic Algorithms (GA) 31
2.5.3.2 Evolution Strategies (ES) 33
2.5.3.3 Evolutionary Programming (EP) 35
2.5.3.4 Genetic Programming (GP) 35
2.5.3.5 Comparing GA and GP 37
2.5.3.5.1 Representation of Genotypes 37
2.5.3.5.2 Complexity of Interpretation 38
2.5.3.5.3 Syntax Preserving Crossover 39
2.6 When is a Weak Method Not Weak? 40

III THE EVOLUTIONARY WEAK METHOD AND EMERGENT INTELLIGENCE 41

3.1 Empirical Credit Assignment 41
3.1.1 Representation Specific Operators 42
3.1.2 Abstract Feature Propagation 43
3.1.2.1 Schema Theory 45
3.1.2.2 Abstract Features 46
3.1.3 The Strength of Reproduction 50
3.2 Emergent Intelligence 51
3.2.1 Emergence and Emergent Computation 51
3.2.1.1 Emergence and Emergent Properties 53
3.2.1.2 Emergent Computation 55
3.2.2 Emergent Intelligence: The Very Idea 56
3.2.2.1 Necessity of Interaction with the Task Environment 57

3.2.2.2 Task-Specific vs Representation-Specific Knowledge 57
3.2.2.3 The Role of the Observer in Emergent Intelligence 57
3.2.2.4 Non-referential Symbols and Representations 58
3.2.2.5 A Comparison Between EI and AI 58
3.2.3 Review of Related Approaches 60
3.2.3.1 Society of Mind 60
3.2.3.2 Embodiment 60
3.2.3.3 Situated Action 61
3.3 Innate Emergent Intelligence in Genetic Programming 62
3.3.1 Emergence of Introns 62
3.3.2 Emergent Diploidy and Dominance 63
3.4 Demonstrations of Emergent Intelligence 64

IV THE EMERGENCE OF TASK-SPECIFIC STRUCTURES 65

4.1 Connectionist Architectures 65
4.2 The Complete Network Induction Problem 68
4.3 Evolving Connectionist Networks 69
4.3.1 Evolving Networks with Genetic Algorithms 69
4.3.2 Network Induction with Evolutionary Programming 73
4.4 The GNARL Algorithm 75
4.4.1 Selection, Reproduction and Mutation of Networks 76
4.4.1.1 Severity of Mutations 78
4.4.1.2 Parametric Mutation of Networks 78
4.4.1.3 Structural Mutation of Networks 79
4.4.2 Fitness of a Network 80
4.5 Experiments. 81
4.5.1 Inducing Regular Languages 81
4.5.2 The Ant Problem 86
4.6 Discussion 94
4.7 Conclusion 95

V THE EMERGENCE OF TASK-DIRECTED COMPONENT MANIPULATION 96

5.1 Evolving Finite State Machines 96
5.2 Freezing and Unfreezing Representational Components 98
5.3 Experiments 99

5.4 Results and Discussion 100
5.5 Conclusion 103

VI THE EMERGENCE OF MODULAR SOLUTIONS AND HIGH-LEVEL REPRESENTATIONS 104

6.1 Problem Decomposition 104
6.2 The Genetic Library Builder 106
6.2.1 Compression of Modules 107
6.2.2 Expansion of Compressed Modules 110
6.2.3 Non-monotonic Formation of Hierarchical Decompositions 110
6.2.4 Evaluation of Modules 111
6.2.5 The Emergence of High-Level Representations 112
6.2.6 Task-Specific Feature Acquisition 113
6.3 Generality of Emergent Modules 114
6.4 Comparison to Koza's ADFs 114
6.5 The Emergence of Environment Specific Modules 115
6.5.1 Tower of Hanoi Experiments 115
6.5.2 Experiments with Tic Tac Toe 117
6.6 Discussion 123
6.7 Conclusion 123

VII EMERGENT GOAL-DIRECTED BEHAVIOR 125

7.1 Competitive Learning 126
7.2 Competitive Learning in Evolutionary Algorithms 128
7.2.1 Standard Fitness Functions and Competitive Selection 129
7.2.2 Competitive Fitness Functions 130
7.2.2.1 Axelrod's Full Competitive Model 130
7.2.2.2 Hillis' Bipartite Competitive Model 132
7.3 Tournament Fitness 134
7.4 Experiments 136
7.5 Results 138
7.6 Discussion 141
7.7 Conclusion 142

VIII SUMMARY AND CONCLUSIONS 144

BIBLIOGRAPHY 152

LIST OF TABLES

TABLE **PAGE**

1. Differences between artificial intelligence and emergent intelligence. 59
2. Characterizations of the seven languages from Tomita (1982). 81
3. Positive and negative examples for the seven languages investigated in Tomita (1982). ε signifies the null string. 83
4. Results from Watrous and Kuhn (1992) for the seven languages shown in Table 3. 84
5. Experimental results for GNARL on the seven languages shown in Table 3. Units are identical for the same measures in Table 4. 85
6. Speed-up results for FSA induction for 4 runs with and without freeze and unfreeze. 101
7. Table showing results of competition experiments. 138

LIST OF FIGURES

FIGURE **PAGE**

1. Comparison of the effectiveness of weak and strong methods over the variety of tasks that can be attempted. Strong methods are specialized for a set of tasks and more effective while weak methods are general across tasks but less effective. 18
2. Algorithm for Generate and Test weak method. 19
3. Algorithm for hill climbing weak method. the switch function switches the values of its arguments so that each has the others value. 20
4. Recursive algorithm for depth-first search. O is the set of operators that defines the connectivity of the search space. The connectivity of the space is assumed to be a tree. 21
5. Algorithm for beam search. *obj-queue* and *next-queue* are standard LIFO queues with standard operators *head*, *push* and *pop*. The sort routine is assumed to use Tester to determine the ordering so that the objects closest to being solutions are at the front of the queue. The space is assumed to be a tree. 22
6. Generic algorithm for evolutionary algorithms. 26
7. The separation of structure and behavior. (a) Emphasis on acquiring structure results in creating offspring that are structurally similar but may be behaviorally incomparable. (b) Emphasis on acquiring behaviors might require operators that associate dissimilar structures. 28
8. Operators used by genetic algorithms. (a) single point crossover; (b) point mutation; (c) inversion. 32
9. Structure of a perceptron. The binary retina feeds into a set of masks that essentially multiply their inputs. These binary variables are then multiplied by the weights and summed. If the sum is larger than the stored threshold, then the concept represented by the preceptron appears in the retina. 66

10. Standard connectionist architecture. *x1* and *x2* are inputs to the network and are set by the environment. Their activation is multiplied by the weight on the associated connection. A hidden or output node sums all inputs, subtracts the threshold which is shown as a value inside the node, and applies a sigmoid activation function to determine output. The state of the output node is the The above network computes the exclusive-or of the inputs. 67

11. The dual representation scheme used in genetic algorithms. The interpretation function maps between the elements in recombination space on which the search is performed and the subset of structures that can be evaluated as potential task solutions. 70

12. The competing conventions problem (Schaffer, Whitley and Eshelman 1992). Bit strings A and B map to structurally and computationally equivalent networks that assign the hidden units in different orders. Because the bit strings are distinct, crossover is likely to produce an offspring that contains multiple copies of the same hidden node, yielding a network with less computational ability than either parent. 72

13. The evolutionary programming approach to modeling evolution. Unlike genetic algorithm, evolutionary programs perform search in the space of networks. Offspring created by mutation remain within a locus of similarity to their parents. 74

14. Sample initial network. The number of input nodes (min) and number of output nodes (mout) is fixed for a given task. The presence of a bias node (b = 0 or 1) as well as the maximum number of hidden units (hmax) is set by the user. The initial connectivity is chosen randomly (see text). The disconnected hidden node does not affect this particular network's computation, but is available as a resource for structural mutations. 77

15. The number of network evaluations required to learn the seven data sets of Table 3. GNARL (using both SAE and SSE fitness measures) compared to the average number of evaluations for the five runs described in Watrous and Kuhn (1992). 86

16. Percentage accuracy of evolved networks on languages in Table 3. GNARL (using SAE and SSE fitness measures) compared to average accuracy of the five runs in Watrous and Kuhn (1992). 87

17. The ant problem. Black squares indicate food while grey squares show the shortest path. The trail is connected initially, but becomes progressively more difficult to follow. The underlying 2-d grid is toroidal, so that position "A" is the first break in the trail – it is simple to reach this point. Positions "B" and "C" indicate the only two positions along the trail where the ant discovered in run 1 behaves differently from the 5-state FSA of Jefferson et al. (1992) (see Figure 20). 88

18. The semantics of the I/O units for the ant network. The first input node denotes the presence of food in the square directly in front of the ant; the second denotes the absence of food in this same square. This particular network finds 42 pieces of food before spinning endlessly in place at position P, illustrating a very deep local minimum in the search space. 89

19. The Tracker Task, first run. (a) The best network in the initial population. Nodes 0 and 1 are input, nodes 5-8 are output, and nodes 2-4 are hidden nodes. (b) Network induced by GNARL after 2090 generations. Forward links are dashed; bidirectional links and loops are solid. The light gray connection between nodes 8 and 13 is the sole backlink. This network clears the trail in 319 epochs. 90

20. FSA hand-crafted for the Tracker task by Jefferson et al. (1992). Food is indicated by 1, no food is indicated by 0. This simple system implements the strategy "move forward if there is food in front of you, otherwise turn right four times, looking for food. If food is found while turning, pursue it, otherwise, move forward one step and repeat." This simple FSA traverses the entire trail in 314 steps, and gets a score of 81 in the allotted 200 time steps. 91

21. Limit behavior of the network that clears the trail in 319 steps. Graphs show the state of the output units Move, Right, Left. (a) Fixed point attractor that results for sequence of 500 "food" signals; (b) Limit cycle attractor that results when a sequence of 500 "no food" signals is given to network; (c) All states visited while traversing the trail; (d) The path of the ant on an empty grid. The z axis represents time. Note that x is fixed, and y increases monotonically at a fixed rate. The large jumps in y position are artifacts of the toroidal grid. 92

22. Limit behavior of the network of the second run. Graphs show the state of the output units Move, Right, Left. (a) Fixed point attractor that results for sequence of 3500 “food” signals; (b) Limit cycle attractor that results when a sequence of 3500 “no food” signals is given to network; (c) All states visited while traversing the trail; (d) The path of the ant on an empty grid. The *z* axis represents time. The ant’s path is comprised of a set of “railroad tracks.” Along each track, tick marks represent back and forth movement. At the junctures between tracks, a more complicated movement occurs. There are no artifacts of the toroidal grid in this plot, all are actual movements (cf. Figure 21d). 93

23. FSM evolved with compression in 420 generations. Frozen states and transitions are shown in grey. Input symbol set is (F, N) and output symbol set is (M, L, R, N) as described in the text. Extraneous states and transitions are not shown. The initial state is indicated by the oversized arrow. 102

24. Creation of a compressed module from a randomly selected subtree of a genotype. Newfunc’s definition is added to the library. 107

25. Depth compression applied to a subtree larger than the maximum depth. The module’s definition has parameters as a result. 108

26. Leaf compression applied to a selected subtree.ote that identical leaves are represented by the same parameter name in the defined function. Expansion is the reverse of this process. 109

27. Primitive language for Tower of Hanoi experiments. Each command moves a single disk between the specified pegs. 115

28. Evolved program for Towers of Hanoi experiment. Compressed modules are designated with the name FUNC####. 116

29. Sequence of moves implemented by an evolved module. The configuration of the disks in Step 1 is the configuration at which the module is called in the evolved program. 117

30. Primitives used to evolve modular programs to play Tic Tac Toe. 118

31. Best evolved modular program from described experiment. Evolved modules are named FUNC####. 120

32. A sample game from the described run. The evolved program is "X" and is playing first. It sets up the first fork in (b) and completes it in (c). The second fork is completed in (d) and leads to a win in (e). The evolved program received a score of 20 points for this game. 121

33. Graphs showing number of calls (y-axis) per generation (x-axis) for three evolved modules. Note the scales are different in each. 122

34. Three types of competitive fitness functions. (a) Full competition used in Axelrod (1989); (b) Bipartite competition used in Hillis (1992); and (c) Tournament fitness with each horizontal line designating a competition and each upward arrow designating the winner progressing in the tournament. 131

35. Number of underdogs that won their competition for each generation of the run. 140

36. Average (solid) and maximum (dashed) "objective" fitness difference between winning underdogs and their competitor. 141

LIST OF SYMBOLS

SYMBOL	DEFINITION
μ	Any abstract feature of a representation. This includes a specific feature of the representation or a quality of the interpretation of the representation.
H	Holland's (1975) symbol for a particular type of feature called a schema.
$m(\mu, t)$	The number of population members with the feature μ at time t.
$f(H)$	The average fitness of all population members with schema H.
$\bar{f}$	The average fitness of the population.
f_{max}	The maximum fitness for a given task.
n	The size of the population.
P_t	The set of objects in the population at time t.
$\varepsilon(\mu, t)$	The probability that feature μ is disrupted by an operator during reproduction.
o_i	The ith object of the population.
$\sigma(o_i, t)$	The probability of selecting object o_i from the population.
$\Lambda(\mu, t)$	The set of objects in the population at time t that posses the feature μ.
$\overline{\sigma}(\mu, t)$	The average probability of selecting a member of $\Lambda(\mu, t)$ from the population.
η	A connectionist network.

x_i	The *i*th input to a connectionist network.
θ_i	The threshold associated with node *i* in a connectionist network.
$T(\eta)$	The temperature of a network.
α	A user defined proportionality constant.
$\hat{T}(\eta)$	The instantaneous temperature of a connectionist network.
Δ_{min}	The minimum number of structural mutations made to an offspring.
Δ_{max}	The maximum number of structural mutations made to an offspring.
$\lambda(i)$	The maximum number of links in a GNARL network with *i* hidden nodes.
$Out(\eta, x_i)$	The output response of network η given input x_i.
$\varepsilon(\eta)$	The error of the network over the given task.
$N(\mu, \sigma)$	A gaussian random variable with mean $\mu$and standard deviation σ.
$U(0,1)$	A uniform random variable over the interval (0, 1).