Foundations of evolutionary computation

David B. Fogel*
Natural Selection, Inc., 3333 N. Torrey Pines Ct., Suite 200, La Jolla, CA, USA 92037

ABSTRACT

Evolutionary computation is a rapidly expanding field of research with a long history. Much of that history remains unknown to most practitioners and researchers. This paper offers a review of selected foundational efforts in evolutionary computation. A brief initial overview of the essential components of evolutionary algorithms is presented, followed by a review of early research in artificial life, evolving programs, and evolvable hardware. Comments on theoretical developments and future developments conclude the review.

Keywords: evolutionary computation, artificial life, evolvable programs, evolvable hardware

1. INTRODUCTION

The essential aspects of an evolutionary algorithm are shown in Figure 1. A population of candidate solutions to a problem is scored with respect to a so-called fitness criterion. These solutions serve as parents for offspring. These offspring are created via random variation of the parents in the form of mutations and/or recombination, or possibly other operations. The offspring are scored and the solutions compete for survival and the right to become parents of the next generation. This basic protocol or variants of it appear in virtually all evolutionary algorithms, with the exception of artificial life studies that seek to determine emergent properties of simulations, and also some extended variations of evolutionary computing such as particle swarm and ant colony methods. In the past 20 years, there have been numerous successful applications of this general evolutionary approach, with various extensions, in the areas of medicine, bioinformatics, military planning, scheduling, forecasting, and other areas. It is reasonable to believe that over 3000 papers are published annually in evolutionary computing around the world.

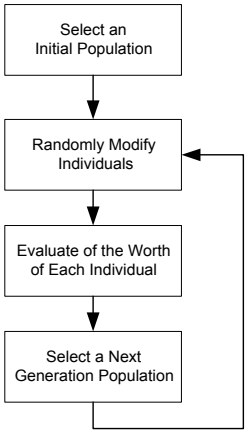

Figure 1. The basic flow of an evolutionary algorithm.

Evolutionary computation has a long history, which extends over 55 years. Some of the earliest progenitors of that history can be found even 20 years still earlier. For example, in 1932 Cannon[1] noted that evolution

* dfogel@natural-selection.com; phone 1 858 455-6449; fax 1 858 455-1560; www.natural-selection.com

Modeling and Simulation for Military Applications, edited by Kevin Schum, Alex F. Sisti
Proc. of SPIE Vol. 6228, 622801, (2006) · 0277-786X/06/$15 · doi: 10.1117/12.669679

was a learning process and made a direct comparison to individual learning. In addition, Turing[2] offered that there is an "obvious connection between [machine learning] and evolution." By the mid-1950s, the idea of simulating evolution on a computer had already taken root. One of the first theoretical works on simulated evolution involved a robot learning to move in an arena. In this work, Friedman[3,4] also remarked that mutation and selection would be able to design thinking machines, including chess-playing machines. The concept that evolution and learning are intimately connected was also offered by Campbell[5,6,7] in the late 1950s and also 1960.

Unfortunately, these historical contributions, and likely the majority of other early contributions, are virtually unknown in the evolutionary computation community. This unawareness is a result of a lack of rigor and a reflection of the immaturity of the field, as compared with say, physics, mathematics, or chemistry. Many innovative approaches to evolutionary computing were undertaken in the earliest days of computing. With current computational capabilities, we now have the computing power to explore these seminal ideas further. If this review paper encourages such efforts, it will have served its purpose. This paper focuses on historical contributions in artificial life, modeling genetic systems, evolving programs, and evolving hardware which may not be as well known as others in popular literature (e.g., those featured routinely in trade magazines). It also offers remarks on the future of evolutionary algorithms in light of current knowledge. Not all early and unrecognized efforts are given due attention owing to space limitations. Interested readers may find these covered in Fogel[19].

2. HISTORICAL FOUNDATIONS

1.1 Artificial Life

As noted in Fogel[8,13], perhaps the earliest published record of any work in evolutionary computation is Barricelli[9]. Nils Aall Barricelli worked on John von Neumann's high-speed computer at the Institute for Advanced Study in Princeton, New Jersey, in 1953. Barricelli's experiments were essentially trials in the area of artificial life, in which numbers were placed in a grid and moved based on local interaction rules. His original research was published in Italian, but was republished in 1957 in English[10]. Two additional publications in 1962 and 1963 extended his work[11, 12].

Barricelli's essential experiments worked as follows. Numbers were entered into a grid of predetermined size. Positive numbers shifted to the right, while negative numbers shifted to the left. When collisions occurred between two more numbers that entered the same cell in the grid, other rules were applied to determine how to alter the numbers. For example, suppose that a one-dimensional grid had $N = 20$ cells and numbers were distributed in those cells at the initial generation $g = 0$. For $g = 1$, the numbers would shift to different cells based on the arrangement of numbers at $g = 0$, and then progress further or "migrate" based on their then-current positions.

To explain Barricelli's specific migration rules, say $x_{i,g}$ is the numeric entry at the gth generation in cell i. Barricelli [2] used rules such as:

(1) A number shifts n cells to the right if it is positive, or $|n|$ cells to the left if it is negative, unless this results in a "collision" with another number (in which two numbers arrive at the same cell location).

(2) The same number $x_{i,g} = n$ may reproduce m cells to the right (or left) if $x_{i+n,g} = m$, again with the exception of a collision.

(3) Reproduction may occur more than once if $x_{i+m,g} = r$ (where $r \neq 0$, which designated an empty square), then $x_{i+r,g+1} = n$, again with the exception of a collision.

(4) If two numeric elements collide in a cell, then if they are equal only one copy of the number is placed in the cell. If the numbers are not equal, other rules are applied to determine which number to place in the cell or other cells.

Figure 2, taken from Barricelli[10], shows an example of numbers propagating in time through a series of 20-cell grids. The numbers involved are −3, 1, and 5, and also 0 representing an empty square. Starting from the arrangement at $g = 0$ at the top of the grid (usually found by using a set of playing cards), by the fourth generation, the pattern (5, −3, 1, −3, 0, −3, 1) appears and persists in every other generation. This example is a "flat" grid, but Barricelli[10] noted experiments with 512 cells in a tubular design (connecting the left and right edges).

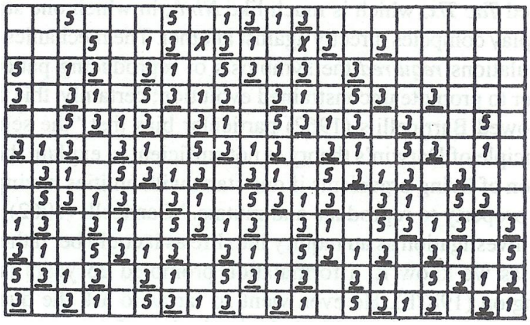

Figure 2. A series of generations of numbers moving in a grid following the rules of Barricelli[10]. The first generation is shown in the top row of cells. Subsequent generations are shown as successive descending rows of cells. By the fourth generation, the pattern (5, −3, 1, −3, 0, −3, 1) appears and persists in each subsequent generation. Note that a number with an underscore has a negative value. The figure is adapted from Barricelli[10].

Barricelli made several observations about the patterns that emerged from such simple rules, which he termed "organisms." Organisms were defined to be *independent* if they could reproduce without requiring other organisms of a different pattern, which he described using the term "another species." Conversely, if an organism was unable to reproduce without a continuous supply of other numbers then it was *dependent*, and described as a "parasite." Barricelli noted patterns of recombination, including a multiple-point operation in which two patterns would collide and the result would be a new self-sustaining pattern with numbers chosen from each of the "parents." A two-dimensional version of the experiment was also performed, with obvious similarities to the work of Conway's Game of Life, popularized much later[14]. Overall, Barricelli's search for emergent patterns is reminiscent of the search for emergent properties in complex adaptive systems that pervaded artificial life research in the late 1980s and early 1990s[15,16].

In the late 1960s, Michael Conrad offered a seminal contribution to artificial life[17], the journal version of which was published by Conrad and Pattee[18]. A population of cell-like individual organisms was subjected to a strict materials conservation law that induced competition for survival. The organisms were capable of mutual cooperation as well as executing biological strategies that included genetic recombination and the modification of the expression of their genome. No fitness criteria were introduced explicitly as part of the program. Instead, the simulation was viewed as an ecosystem in which genetic, individual, and population interactions would occur and behavior patterns would emerge.

The organisms were composed of what were essentially genetic subroutines. Individual phenotypes were determined by the manner in which these routines were used by the organisms. The fixed set of routines that they provided was somewhat limiting, but Conrad and Pattee[18] accepted this limitation so long as the "process of modifying these featured produces a system whose behavior converges to that of a natural ecosystem." In particular, Conrad and Pattee were concerned that (1) behavior that is characteristic of ecological succession processes should potentially emerge, (2) the processes of evolutionary search must agree with biological fact (i.e., nonbiological operators that would make search more "efficient" were precluded), and (3) the simulation should be as simple as possible to allow for studying both the fundamental features of ecosystems as well as the minimum necessary conditions for natural evolution.

The events in the evolution program took place in an environment termed the *world* (Figure 3). The world was a one-dimensional string of *places*. The first element in this array was linked to the last, forming a loop, to avoid end effects. Each place in the world was characterized by a state (A or B) and a certain number of material parts called *chips*. The organisms determined the control of the flow of the chips in the world. Chips were essential to persistence (i.e., survival) and reproduction, but no explicit behavioral strategies were introduced to acquire these chips. Any such behaviors emerged from the initial conditions of the simulation as the conservation of matter (a fixed number of total chips) induced a competition for chips among the organisms. Figure 4 shows a flowchart of the overall process.

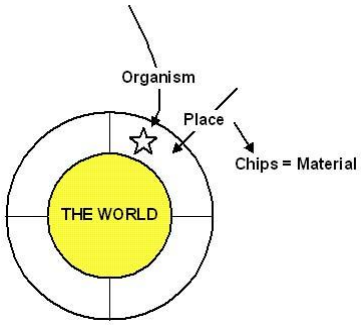

Figure 3. A graphical depiction of Conrad's Evolve "world." Organisms resided in places in the world, which was a ring of places. Chips were located in places and served as the material that organisms required.

Each organism in the simulation was associated with a place in the world. Organisms operated over a range of contiguous places (territory). Events occurred in discrete periods with the lifespan of an organism extending over a number of these periods. Organisms interacted in a two-phase system: In the first phase they interacted with the environment and the other organisms in their local areas, while in the second phase chips were collected for the various behaviors, reproduction occurred (if applicable), and chips from "dead" organisms were deutilized (i.e., returned to a "matter pool"). The effects of birth and decay determined the composition of the next iteration, and the process was then repeated.

As indicated above, the organisms were modeled with an explicit genotype-phenotype. An organism's genome was mapped to a phenotype according to a coding function. Sixteen possible pairs of genomic symbols were associated with six types of phenome symbols. The organism's immediate behavior depended on an "internal state" and an "input state." The sequence of internal states, combined with the input states and the other organisms in the population, determined the chip-collecting behaviors of the organism. Organisms could allocate chips for self-repair, and reproduction was allowed when a sufficient number of chips had been stored. The genetic program of the organisms was subjected to both point and size mutations, as well as a recombination operator that consisted of breaking and splicing genomes at random places. The location of daughter cells created by reproduction was restricted, but was also under genetic control.

The population size ranged from 200 to 400 organisms, depending on the number of chips made available at the start of the simulation. The distribution of chips was affected only by the organisms' behavior and therefore the fitness criteria varied during the simulation as organisms interacted (noise was also added to the environment in some experiments). Attention was focused on features such as chip utilization (ratio of chips used to chips not used), matching ratio (successful matches to those attempted), and changes in size of population (survival curve). A sample result and the associate figure caption from Conrad's primary work is shown in Figure 5.

Some general observations of these experiments included: (1) the matching ratio of phenotypic characters to environmental conditions increased when the environment contained no noise, (2) adding noise led to a diversity of phenotypic types with no indication of impending homogeneity, (3) in general, the probability of recombination tended to decrease, (4) utilization of the environment tended to increase, and (5) the

predominant organisms carried phenome sequences that were not executed and therefore of no selective value. These initial simulations were extended over many years in Conrad[20], Conrad and Strizich[21], Rizki and Conrad[22,23], Conrad and Rizki[24], and O'Callaghan and Conrad[25].

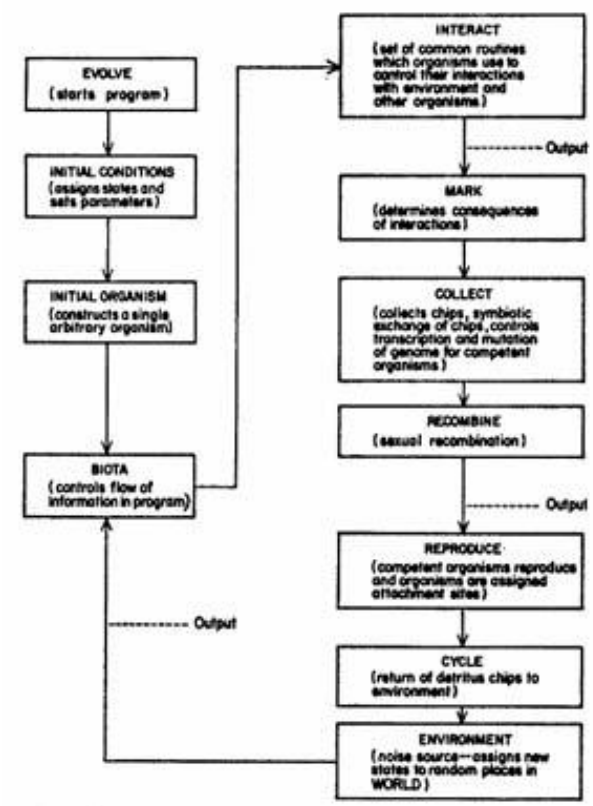

Figure 4. The flowchart for Conrad's Evolve program.

WORLD	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
INPUT STATE	A	A	B	A	B	B	B	A	B	A	A	A
INTERNAL STATE		A	C	B	D	F	E	AB	AA	E	D	
		ORGANISM										

FIG. 2. Representative organism. The organism is attached at place P2. The first internal state is A and the first input state is A. The organism records this fact, and will collect a certain number of chips during the second pass. The second internal state is C. The organism will later look for a conjugate at position P3. The third internal state is B and the input state is A. These do not match and the organism goes to its next internal state. This is D, and the organism can allocate a chip to repair processes, assuming that it collects a chip. The fifth internal state is F, and the organism will later look for a symbiont at place P6. The next internal state is E and the organism goes into a parametric mode. The seventh internal state is one of the pair A or B. Thus the organism will always match place P8. The internal state which follows is also one of a pair, but in this case both members of the pair are A's. This is wasteful and, in fact, the organism does not match place P9. The organism now goes into an E state and jumps out of the parametric mode. Thus the next state is D and the organism is entitled to another repair chip. In general, if the second alternative in the parametric mode is a non-matching type symbol it cannot be used. Also it is wasteful if the E symbol signalling the parametric mode occurs in an odd position, since the immediately following symbol will not be used.

Figure 5. A representative organism's behavior from Conrad's Evolve model and associated description of the program's activity in the caption.

Conrad's investigations were clear precursors to the later work of Ray[26] with Tierra, in which assembly language programs competed for CPU cycles. In each case, no explicit fitness criteria were applied to direct the course of evolution.

1.2. Modeling Genetic Systems

The interest in modeling genetic systems to explore and better understand biology dates back at least to the work of Fraser[27] in 1957, who conducted simulations involving diploid organisms represented by binary strings of a given length (*n* bits). Each bit represented an allele (dominant or recessive) and the phenotype of each organism was determined by its genetic composition. Reproduction was accomplished using an *n*-point crossover operator in which each position along an organism's genetic string was assigned a probability of breaking for recombination. *Linkage groups* could form between genes. This was accomplished by varying the probability of crossover occurring at each locus along either string.

Fraser's general procedure used a population of *P* parents that generated *P′* offspring via recombination. Selection then eliminated all but *P* of the offspring (and all of the parents were also eliminated). Readers who are well versed in evolution strategies will note that this anticipated the (μ,λ) selection method. Selection could be applied toward extreme values of a phenotype (the equivalent of function maximization or minimization) or the mean values (stabilizing selection against extremes). The possibility for varying the number of progeny per parent was also introduced.

Fraser and colleagues published numerous papers in this line of research. These works included Fraser[28,29,30,31,32,33], Barker[34,35], Fraser and Hansche[36], Fraser et al.[37], Fraser and Burnell[38,39]. Fraser and Burnell[40], *Computer Models in Genetics*, was the second book in the history of evolutionary computation. Given the close overlap in subsequent work in genetic algorithms, as say offered in Holland[41], including epistasis, linkage groups, recombination, binary strings, diploid representations, and so forth, it is perhaps surprising and certainly unfortunate that these earlier works were mostly ignored outside the field of genetics for over two decades. By the time of Fraser's later contributions, say Fraser[42], the models also incorporated inversion to build arbitrary linkages between alternative genes and were described in clear terms as "purposive" and "learning" (Figure 6). Thus Fraser's model was essentially equivalent to the canonical genetic algorithm popularized in Holland[41].

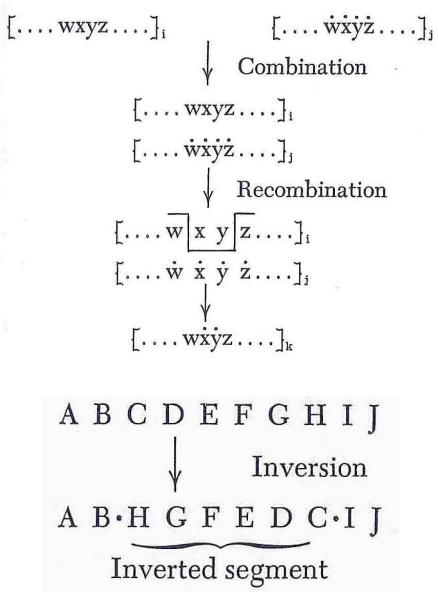

Figure 6. Taken from Fraser[42], the top portion depicts a 2-point crossover operation, and the bottom portion depicts inversion on labels of genes in a string.

Whereas Fraser did not offer mathematical theory about his algorithms, Bremermann studied a similar model and did make theoretical contributions. In 1958, Bremermann[43] offered an idealized model of

evolution in which heritable genes were encoded in binary strings that were processed by reproduction (sexual or asexual), selection, and mutation. Bremermann showed that for the *onemax* problem, in which fitness is determined by summing the number of 1s that occur in a binary string, the most favorable mutation probability for improvement without degeneration when the number of incorrect bits is small with respect to the total number of bits, n, is approximately equal to $1/n$. This result was rediscovered over three decades later in Muehlenbein[44]. Bremermann also derived the generalized optimum probability of mutation for this problem as $1 - (m/n)^{1/(n-m)}$, where m is the number of bits that are correct. Bremermann et al.[45] extended the result to hold for any fitness function that is a function of n binary variables and is a monotone function of the Hamming distance of its argument vector from the solution vector.

Bremermann and colleagues extended this original model to become a generalized function optimization method in Bremermann[46,49,50,51,52], Bremermann and Rogson[47], and Bremermann et al.[45,48]. The method was applied in numerous experiments to linear and nonlinear optimization, including problems of up to 30 dimensions. Bremermann considered various recombination operators, including the potential for using more than two parents (Figure 7).

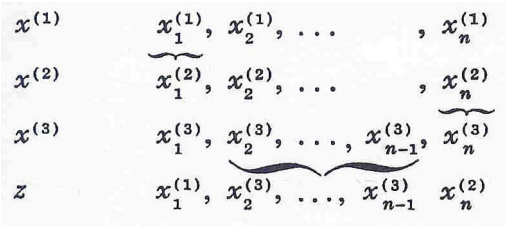

Figure 7. Bremermann's studies of recombination included the potential for creating an offspring from two or more (in this example, three) parents (from Bremermann et al.[45]).

1.3. Evolving Programs

Conrad's artificial ecosystem experiments described earlier involved organisms that were programs, which reacted to inputs, followed a sequence of steps, and generated output; however, two other earlier efforts were made to evolve programs to execute specific tasks.

In 1958 and 1959, Friedberg[53] and Friedberg et al.[54] evolved machine language programs (binary language) to perform simple operations such as moving data from one location to another or performing the sum of numbers in two data locations. The speed of the available computers at the time demanded several procedural shortcuts in order to test the devised methods. There was a hope that structurally similar programs could be grouped together in *classes*. This concept was another forerunner of what was described as *intrinsic parallelism* in Holland[41]. Specifically, a class was defined to consist of all programs having a certain instruction at a certain location. A learning procedure was intended to compare the performance of two nonoverlapping classes of programs, essentially evaluating alternative instructions that might occur at a given location. This concept is also similar to what was later described as *schemata* in Holland[41].

A credit assignment method was invented to partition the influence of individual instructions, which recorded the number of overall successes and failures for each instruction. The hope was that by combining programs with more "good" instructions this would lead to improved programs overall. Friedberg et al.[54] noted that the evolved programs compared favorably to completely random generate-and-test methods, but complex problems that were broken in parts were not always solved to completion. Misky[55] offered an unfavorable and apparently erroneous criticism of Friedberg's work, saying that "The machine did learn to solve some extremely simple problems. But it took on the order of 1,000 times longer than pure chance would expect." There appears to be no basis for this description. Unfortunately, Minksy's criticism was well publicized and left a lasting impression of this work.

Subsequent to Friedberg, Fogel[56] offered the idea of simulating evolution for creating artificial intelligence, following earlier development of this concept in 1960 while serving at the National Science Foundation.

Fogel suggested that prediction is a keystone of intellect and his research focused therefore on predictive models, taken in the form of finite state machines (Figure 8). A finite state machine is a program that starts in a state and receives an input symbol and generates output and transitions to other states based on a stream of such inputs.

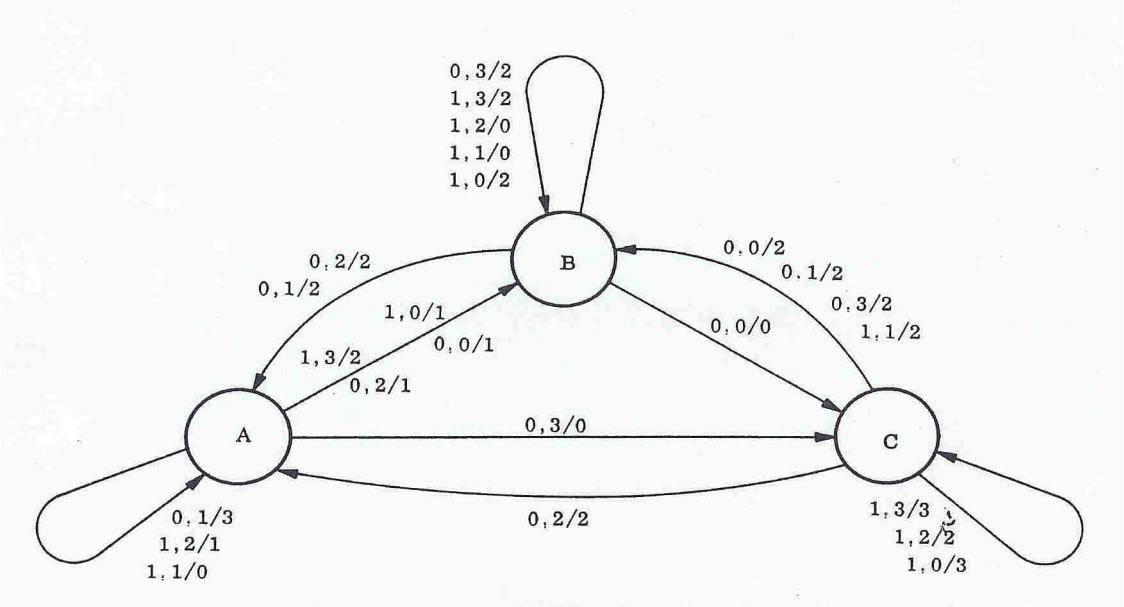

Figure 8. A finite state machine from Fogel et al.[57]. The three-state machine is presumed to start in state A. The machine is designed to operate in a four-symbol environment and can handle two inputs simultaneously in the form of an ordered pair of symbols [e.g., (0,1)]. This could represent, for example, moves in a game by two players. The symbols to the right of the virgules indicate the output associated with possible inputs.

A population of random finite state machines was exposed to an environment as a string symbols (e.g., 101110011101 or 31022210), with each machine being scored in terms of how well it predicted the available data. The better machines were retained as parents for generating offspring via mutation of the number of nodes in the machine, the input-output or input-transition relationships, or the start state. The offspring were then compared with the parents in terms of performance and the process of variation and selection was continued until a prediction of the next symbol was required. At that point the best machine was used to make the prediction and the new datum was added to the observed environment, with the process iterating.

This general procedure, which included prescriptions for using probabilistic selection and multiparent mating, was expanded and applied to problems in prediction, system identification, control, and pattern recognition in a series of studies (Fogel[58], Fogel et al.[59,60,61], Fogel and Burgin[62], Burgin[63,64], Lutter and Huntsinger[65], Cornett[66], Dearholt[67], Atmar[68], and many others).

The publication of Fogel et al.[69] was the first book in the field of evolutionary computation. The evolutionary approach to artificial intelligence received significant attention from the artificial intelligence community, as well as researchers in the former Soviet Union[70,71,72]. Notable resistance was offered from those following more traditional avenues of symbolic processing and heuristics[73].

1.4. Evolvable Hardware

Recent interest in evolvable hardware has focused on evolving electronic circuits (which was suggested in Atmar[68]; however, perhaps the earliest effort in evolvable hardware evolved physical devices such as bent pipes, airfoils, and flashing nozzles. This research was performed by Ingo Rechenberg, Hans-Paul Schwefel, and Peter Bienert in 1964 and subsequently at the Technical University of Berlin, Germany. Instead of using conventional gradient methods to optimize these devices for various tasks, e.g., to design

an airfoil that minimizes drag, they used a random variation and selection method that employed throwing dice to make physical alterations to devices. This was published first in 1965 by Rechenberg[74]. In 1966, Schwefel[75] used this "evolution strategy" to design a nozzle for convergent-divergent flow of hot water that would offer maximum efficiency. He built a nozzle out of brass rings of varying diameter and the concatenated them into a single device. By throwing dice, he changed both the number of rings in the nozzle as well as the diameter. Starting with a conventional design shown in Figure 9, through 45 iterations of variation and selection, the final design shown in the figure was obtained. These efforts led to many other developments in evolution strategies for numerical optimization, including the use of a population of solutions, the common $(\mu+\lambda)$ and (μ,λ) forms of selection, self-adaptation, and various forms of recombination.

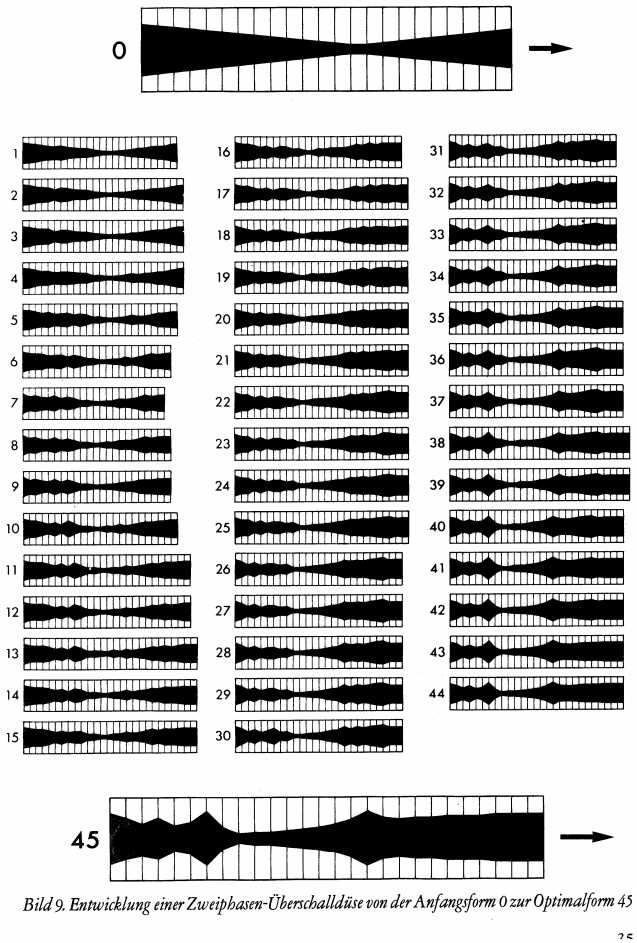

Figure 9. A series of 45 iterations of improvements from Hans-Paul Schwefel's experiments evolving a flashing nozzle. Unlike the initial standard design, the final design did not use a single pinch point.

3. FUTURE DEVELOPMENTS

The field of study of evolutionary algorithms has now reached a critical mass. Only 20 years ago, it was easy to know virtually everyone researching evolutionary algorithms (then described as evolution strategies, evolutionary programming, or genetic algorithms, and then later genetic programming)

personally. Today, this is impossible. Within the last 10 years, there has been sufficient mixing of research ideas and mathematical development to show general convergence properties of evolutionary algorithms[76,77], specific convergence properties on certain funtions[78], as well as rigorous review of earlier theoretical work that has been proven to be incorrect or unfounded (see Wolpert and Macready[79], Macready and Wolpert[80], Fogel and Ghozeil[81,82], and others; for example, claims of optimality of binary representation, single-point crossover, and the *k*-armed bandit analysis in Holland[41]). In light of these and other mathematical analyses, it has become clear that there is no longer a benefit to partitioning evolutionary computation into separate concepts of say genetic algorithms as opposed to evolution strategies. There is no theory that holds for genetic algorithms that would not hold for evolution strategies as they are constituted currently. Thus, the greatest advance of the field of evolutionary computation will come from ignoring such artificial divisions and recognizing the potential for population-based variation and selection broadly.

Evolutionary algorithms are inherently parallel. Yet, to date, a relative minority of applications has taken advantage of this fact. It is not difficult, however, to construct clusters of personal computers and create the equivalent of supercomputing power at relatively low cost. As distributed processing becomes more ubiquitous, so will applications of evolutionary computing.

Evolution is the most ancient primal form of learning. In contrast, research in artificial intelligence has typically passed over investigations into the causative factors of intelligence in order to more rapidly obtain the immediate consequence of intelligence. Such is the case in numerous examples of expert and knowledge-based systems. While these efforts have had success, such as Deep Blue's victory over Garry Kasparov, world chess champion, in May of 1997, they do not advance our understanding of intelligence. They solve problems, but they do not solve the problem of how to solve problems. Evolution provides the solution to the problem of how to solve problems. Darwinian evolution accounts for the behavior of naturally occurring intelligent entities and can be used to guide the creation of artificial entities that are capable of intelligent behavior, going beyond what humans already know, and in the future, will know.

ACKNOWLEDGEMENTS

This review paper contains materials that have appeared in IEEE publications[8,13,19]. As appropriate, these materials are reused or revised and reprinted here in accordance with IEEE copyrights. The author thanks the conference organizers for inviting this review article.

REFERENCES

1. W.D. Cannon, *The Wisdom of the Body*, W.W. Norton, New York, 1932.
2. A.M. Turing, "Computing machinery and intelligence," Mind, Vol. 59, pp. 433-460, 1950.
3. G.J. Friedman, "Selective Feedback Computers for Engineering Synthesis and Nervous System Analogy," Masters Thesis, UCLA, 1956.
4. G.J. Friedman, "Digital simulation of an evolutionary process," *General Systems: Yearbook of the Society for General Systems Research*, Vol. 4, pp. 171-184, 1959.
5. D.T. Campbell, "Adaptive behavior from random response," *Behavioral Science*, Vol. 1, pp. 105-110, 1956.
6. D.T. Campbell, "Perception as substitute trial and error," *Psychol. Rev*., Vol. 63, pp. 330-342, 1956.
7. D.T. Campbell, "Blind variation and selective survival as a general strategy in knowledge-processes," In *Self-Organizing Systems*, M.C. Yovits, S. Cameron (eds.), Pergamon Press, NY, pp. 205-231, 1960.
8. D.B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, 3rd ed., IEEE Press, NY, 2006.
9. N.A. Barricelli, "Esempi numerici di processi di evoluzione," *Methodos*, pp. 45-68, 1954.
10. N.A. Barricelli, "Symbiogenetic evolution processes realized by artificial methods," *Methodos*, Vol. 9:35-36, pp. 143-182, 1957.
11. N.A. Barricelli, "Numerical testing of evolution theories: I. Theoretical introduction and basic tests," *Acta Biotheoretica*, Vol. 16:1-2, pp. 69-98, 1962.

12. N.A. Barricelli, “Numerical testing of evolution theories: II. Preliminary tests of performance symbiogenesis and terrestrial life,” *Acta Biotheoretica*, Vol. 16:1-2, pp. 69-98, 1962.
13. D.B. Fogel, “Nils Barricelli – Artificial life, coevolution, self-adaptation,” *IEEE Computational Intelligence Magazine*, pp. 41-45, February, 2006.
14. A.K. Dewdney, “Computer recreations: The game of life acquires some successors in three dimensions,” *Scientific American*, Vol. 256:2, pp. 16-24, 1987.
15. C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen (eds.) *Artificial Life II*, Santa Fe Institute, Addison-Wesley, Reading, MA, 1990.
16. C.G. Langton, *Artificial Life: An Overview*, MIT Press, Cambridge, MA, 1997.
17. M. Conrad, “Computer experiments on the evolution of coadaptation in a primitive ecosystem,” Ph.D. Diss., Biophysics Program, Stanford, CA, 1969.
18. M. Conrad and H.H. Pattee, “Evolution experiments with an artificial ecosystem,” *J. Theoret. Biol*., Vol. 28, pp. 393-409, 1970.
19. D.B. Fogel (ed.) *Evolutionary Computation: The Fossil Record*, IEEE Press, NY, 1998.
20. M. Conrad, “Algorithmic specification as a technique for computing with informal biological models,” *BioSystems*, Vol. 13, pp. 303-320, 1981.
21. M. Conrad and M. Strizich, “Evolve II: A computer model of an evolving ecosystem,” *BioSystems*, Vol. 17, pp. 245-258, 1985.
22. M.M. Rizki and M. Conrad, “Evolve III: A discrete events model of an evolutionary ecosystem,” *BioSystems*, Vol. 18, pp. 121-133, 1985.
23. M.M. Rizki and M. Conrad, “Computing the theory of evolution,” *Physica 22D*, pp. 83-89, 1986.
24. M. Conrad and M.M. Rizki, “The artificial worlds approach to emergent evolution,” *BioSystems*, Vol. 23, pp. 247-260, 1989.
25. J. O’Callaghan and M. Conrad, “Symbiotic interactions in the EVOLVE III ecosystem model,” *BioSystems*, Vol. 26, pp. 199-209, 1992.
26. T. Ray, “An approach to the synthesis of life,” *Artificial Life II*, C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen (eds.), Addison-Wesley, Reading, MA, pp. 371-408, 1992.
27. A.S. Fraser, “Simulation of genetic systems by automatic digital computers. I. Introduction,” *Australian J. Biological Sciences*, Vol. 10, pp. 484-491, 1957.
28. A.S. Fraser, “Simulation of genetic systems by automatic digital computers. II. Effects of linkage or rates of advance under selection,” *Australian J.. Biological Sciences*, Vol. 10, pp. 492-499, 1957.
29. A.S. Fraser, “Monte carlo analyses of genetic models,” *Nature*, Vol. 181, pp. 208-209, 1958.
30. A.S. Fraser, “Simulation of genetic systems by automatic digital computers, 5-linkage, dominance and epistasis,” *Biometrical Genetics*, O. Kempthorne (ed.), Pergamon Press, NY, pp. 70-83, 1960.
31. A.S. Fraser, “Simulation of genetic systems by automatic digital computers, VI. Epistasis,” *Australian J. Biological Sciences*, Vol. 13:2, pp. 150-162, 1960.
32. A.S. Fraser, “Simulation of genetic systems by automatic digital computers. VII. Effects of reproductive rate and intensity of selection on genetic structure,” *Australian J. Biological Sciences*, Vol. 13, pp. 344-350, 1960.
33. A.S. Fraser, “Simulation of genetic systems,” *J. Theoret. Biol*., Vol. 2, pp. 329-346, 1962.
34. J.S.F. Barker, “Simulation of genetic systems by automatic digital computers. III. Selection between alleles at an autosomal locus,” *Australian J. Biological Science*, Vol. 11, pp. 603-612, 1958.
35. J.S.F. Barker, “Simulation of genetic systems by automatic digital computers. IV. Selection between alleles at a sex-linked locus,” *Australian J. Biological Science*, Vol. 11, pp. 613-625, 1958.
36. A.S. Fraser and P.E. Hansche, “Simulation of genetic system. Major and minor loci,” *Proc. 11th Intern. Cong. On Genetics*, Vol. 3, S.J. Geerts (ed.), Pergamon Press, Oxford, pp. 507-516, 1965.
37. A.S. Fraser, D. Burnell, and D. Miller, “Simulation of genetic systems. X. Inversion polymorphism,” *J. Theoret. Biol*., Vol. 13, pp. 1-14, 1966.
38. A.S. Fraser and D. Burnell, “Simulation of genetic systems. XI. Inversion polymorphism,” *Am. J. Human Genetics*, Vol. 19:3, pp. 270-287, 1967.
39. A.S. Fraser and D. Burnell, “Simulation of genetic systems. XII. Models of inversion polymorphism,” *Genetics*, Vol. 57, pp. 267-282, 1967.
40. A.S. Fraser and D. Burnell, *Computer Models in Genetics*, McGraw-Hill, NY, 1970.
41. J.H. Holland, *Adaptation in Natural and Artificial Systems*, Univ. Mich, Ann Arbor, 1975.

42. A.S. Fraser, "The evolution of purposive behavior," *Purposive Systems*, H. von Foerster, J.D. White, L.J. Peterson, and J.K. Russell (eds.), Spartan Books, Washington D.C., pp. 15-23, 1968.
43. H.J. Bremermann, "The evolution of intelligence. The nervous system as a model of its environment," Tech. Report, No. 1, Contract No. 477(17), Dept. Mathematics, Univ. Washington, Seattle, July, 1958.
44. H. Muehlenbein, "How genetic algorithms really work: Mutation and hill-climbing," *Parallel Problem Solving from Nature 2*, R. Maenner and B. Manderick (eds.), North-Holland, The Netherlands, pp. 15-26, 1992.
45. H.J. Bremermann, M. Rogson, and S. Salaff, "Global properties of evolution processes," *Natural Automata and Useful Simulations*, H.H. Pattee, E.A. Edlsack, L. Fein, and A.B. Callahan (eds.), Spartan Books, Washington D.C., pp. 3-41, 1966.
46. H.J. Bremermann, "Optimization through evolution and recombination," *Self-Organizing Systems-1962*, M.C. Yovits, G.T. Jacobi, and G.D. Goldstein (eds.), Spartan Books, Washington D.C., pp. 93-106, 1962.
47. H.J. Bremermann and M. Rogson, "An evolution-type search method for convex sets," ONR Tech. Report, Contracts 222(85) and 3656(58), Berkeley, CA, May, 1964.
48. H.J. Bremermann, M. Rogson, and S. Salaff, "Search by evolution," *Biophysics and Cybernetic Systems*, M. Maxfield, A. Callahan, and L.J. Fogel (eds.), Spartan Books, Washington D.C., pp. 157-167, 1965.
49. H.J. Bremermann, "Quantitative aspects of goal-seeking self-organizing systems," *Progress in Theoretical Biology*, Vol. 1, Academic Press, NY, pp. 59-77, 1967.
50. H.J. Bremermann, "Numerical optimization procedures derived from biological evolution processes," *Cybernetic Problems in Bionics*, H.L. Oestreicher and D.R. Moore (eds.), Bionics Symp. 1966, Dayton, OH, Gordon and Breach, NY, pp. 543-561, 1966.
51. H.J. Bremermann, "A method of unconstrained global optimization," *Math. Biosciences*, Vol. 9, pp. 1-15, 1970.
52. H.J. Bremermann, "On the dynamics and trajectories of evolution processes," *Biogenesis, Evolution, and Homeostasis*, A. Locker (ed.), Springer-Verlag, NY, pp. 29-37, 1973.
53. R.M. Friedberg, "A learning machine: Part I," *IBM J. Res. Develop*., Vol. 2:1, pp. 2-13, 1958.
54. R.M. Friedberg, B. Dunham, and J.H. North, "A learning machine: Part II," *IBM J. Res. Develop*., Vol 3, pp. 282-287, 1959.
55. M.L. Minsky, "Steps toward artificial intelligence," *Proc. of the IRE*, Vol. 49:1, pp. 8-30, 1961.
56. L.J. Fogel, "Autonomous automata," *Industrial Research*, Vol. 4, pp. 14-19, 1962.
57. L.J. Fogel, A.J. Owens, and M.J. Walsh, "Artificial intelligence through a simulation of evolution," *Biophysics and Cybernetic Systems: Proc. 2nd Cybern. Sciences Symp*., M. Maxfield, A. Callahan, and L.J. Fogel (eds.), Spartan Books, Washington D.C., pp. 131-155, 1965.
58. L.J. Fogel, "On the organization of intellect," Ph.D. dissertation, UCLA, 1964.
59. L.J. Fogel, A.J. Owens, and M.J. Walsh, "On the evolution of artificial intelligence," *Proc. 5th National Symp. Human Factors in Engineering*, IEEE, San Diego, CA, pp. 63-76, 1964.
60. L.J. Fogel, A.J. Owens, and M.J. Walsh, "Intelligent decision making through a simulation of evolution," *IEEE Trans. Human Factors in Electronics*, Vol. HFE-6:1, pp. 13-23, 1965.
61. L.J. Fogel, A.J. Owens, and M.J. Walsh, "Intelligent decision making through a simulation of evolution," *Behavioral Science*, Vol. 11:4, pp. 253-272, 1965.
62. L.J. Fogel and G.H. Burgin, "Competitive goal-seeking through evolutionary programming," Final Report, Contract AF 19(628)-5927. Air Force Cambridge Research Laboratories, 1969.
63. G.H. Burgin, "On playing two-person zero-sum games against nonminimax players," *IEEE Trans. Systems Science and Cybernetics*, Vol. SSC-5, pp. 369-370, 1969.
64. G.H. Burgin, "System identification by quasilinearization and evolutionary programming," *J. Cybernetics*, Vol. 2:3, pp. 4-23, 1974.
65. B.E. Lutter and R.C. Huntsinger, "Engineering applications of finite automata," *Simulation*, Vol. 13, pp. 5-11, 1969.
66. F.N. Cornett, "An application of evolutionary programming to pattern recognition," Master's thesis, New Mexico State University, Las Cruces, NM, 1972.
67. D.W. Dearholt, "Some experiments on generalization using evolving automata," *Proc. 9th Intern. Conf. Systems Science*, Honolulu, HI, pp. 131-133, 1976.

68. J.W. Atmar, “Speculation on the evolution of intelligence and its possible realization in machine form,” Sc.D. dissertation, New Mexico State University, Las Cruces, NM, 1976.
69. L.J. Fogel, A.J. Owens, and M.J. Walsh, *Artificial Intelligence through Simulated Evolution*, John Wiley, NY, 1966.
70. B.S. Fleishman and I.L. Bukatova, “Some analytical evaluations of evolutionary simulation parameters,” *Avtomatika I Vychislitel’naya Teknika*, July-Aug., no. 4, pp. 34-39 [Russian], 1974.
71. B.M. Yakobson, “Modelling evolutionary processes when designing engineering system,” *Pribory I Sistemy Upravleniua*, no. 5, pp. 9-11 [Russian], 1975.
72. M.N. Anisimov and V.G. Indler, “use of evolutionary programming to control nonstationary agricultural objects,” *Elektrotekhnika*, Vol. 59:4, pp. 31-35, 1988.
73. L.J. Fogel and W.S. McCulloch, “Natural automata and prosthetic devices,” *Aids to Biological Communication: Prothesis and Synthesis*, Vol. 2, D.M. Ramsey-Klee (ed.), pp. 221-262, 1970.
74. I. Rechenberg, “Cybernetic solution path of an experimental problem,” Royal Aircraft Establishment, Library Translation 1122, 1965.
75. H.-P. Schwefel, “Experimentelle optimierung einer zweiphasenduese tell 1,” AEG Research Institute Project MHD-Straustrahlrohr 11034/68, Technical Report 35, 1968.
76. D.B. Fogel, “Asymptotic convergence properties of genetic algorithms and evolutionary programming,” *Cybernetics and Systems*, Vol. 25:3, pp. 389-407, 1994.
77. G. Rudolph, “Convergence analysis of canonical genetic algorithms,” *IEEE Trans. Neural Networks*, Vol. 5:1, pp. 96-101, 1994.
78. H.-G. Beyer, *The Theory of Evolution Strategies*, Springer-Verlag, Berlin, 2001.
79. D.H. Wolpert and W.G. Macready, “No free lunch theorems for optimization,” *IEEE Trans. Evolutionary Computation*, Vol. 1:1, pp. 67-82, 1997.
80. W.G. Macready and D.H. Wolpert, “Bandit problems and the exploration/exploitation tradeoff,” *IEEE Trans. Evolutionary Computation*, Vol. 2:1, pp. 2-22, 1998.
81. D.B. Fogel and A. Ghozeil, “A note on representations and variation operators,” *IEEE Trans. Evolutionary Computation*, Vol. 1:2, pp. 159-161, 1997.
82. D.B. Fogel and A. Ghozeil, “Schema processing under proportional selection in the presence of random effects,” *IEEE Trans. Evolutionary Computation*, Vol. 1:4, pp. 290-293, 1997.