人工生命の方法論とその応用

トーマス・エス・レイ
ATR 人間情報通信研究所
ray@hip.atr.co.jp, ray@santafe.edu, ray@udel.edu
http://www.hip.atr.co.jp/~ray/

本論文では，ソフトウェア進化，すなわち，コンピュテーショナルな媒体を用いて進化のプロセスの具現化を試みる研究動向について解説する．本分野での傑出した研究事例やそのための主だった方法論を紹介するとともに，それらの成しえた成果と直面する課題について考察する．

Alife Methodology and its Applications

Thomas S. Ray
ATR Human Information Processing Research Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-02, Japan

Abstract: A review of efforts to implement the process of evolution in the computational medium. The review will cover prominent examples, and discuss the major classes of implementations, their successes and the obstacles they face.

1 Introduction

The first decade of Artificial Life research has revealed evolution to be one of the most rewarding of life processes to implement in artificial media. While most of this work has involved software evolution, there has been some very exciting work with synthetic evolution in the chemical medium [7, 6], and some new developments that may eventually permit the evolution of hardware [16].

While work with evolving systems entirely created by humans goes back only two or three decades, and the currently accepted theory of evolution originated with Darwin and Wallace in 1859, the applied science of evolution predates both by ten to fifteen thousand years. Management of the evolution of other species has been a fundamental aspect of the human condition since the origin of agriculture, and formed the basis of civilization.

At this time humans were already scattered to all inhabitable regions of the world and began to domesticate plants and animals for food and pets. This domestication process, in large part, involves controlling the evolution of the other species. We influence where they live, how many individuals there are in the population, and which individuals reproduce.

Through this process, we have been able to transform the wild products of organic evolution from fairly poor quality food and pets into the very high quality domesticated organisms that we enjoy today: rice, corn, wheat, chicken, pigs, cows, dog, cats, etc. All of these organisms have been dramatically altered from the wild condition in which our ancestors found them, through breeding (the application of fitness functions).

However, this is not the only method that we have used to derive applications from the process of organic evolution. There are many living organisms from which we derive useful products, which we use essentially in their wild state. Mahogany is used for its very high quality wood, and alligators for their leather, with no artificial breeding. Some species fall between: silk moths and minks are farmed, but probably have been subjected to relatively little breeding.

The inoculation of the process of evolution into media created by humans opens a new chapter in our relationship to evolution. In this manuscript, I will review some of those efforts in the area of software evolution, and I will compare them to our analogous relationship to organic evolution.

2 Digital Evolution – Giving up Control

While there have been a variety of trends during the development of our relationship to synthetic evolutions, I wish to focus on our deliberate efforts to reduce our control of the process, thereby setting evolution free, to proceed in a fashion most closely analogous to the way it operated in creating organic life on Earth.

2.1 Genetic Algorithms

Perhaps the earliest and best know work with digital evolution is what has come to be known as genetic algorithms (GA) [5, 3, 2]. Generally, genetic algorithms use bit strings to encode the solutions to some engineering problem. The bits might encode the values of coefficients in some complex equation. By mutating and crossing over bit strings in a large population, and repeatedly evaluating the "fitness" of the solutions, and preferentially replicating the most fit solutions, the genetic algorithm could search for the optimal combination of the coefficients. However, generally, the form of the equation must be determined in advance.

During the design of the GA, the manner in which the bit string encodes the solution space is determined. Normally this involves a fixed length bit string, with successive portions of the string assigned to the representation of predetermined quantities, such as the coefficients in an equation. Thus the form of the solution is determined in advance, and does not take part in the evolutionary process.

Although researchers in the GA field freely use the phrase "natural selection" to describe their evolutionary process, the GA absolutely does not use natural selection. It uses artificial selection. The designer of the GA writes a "fitness function" algorithm, which determines which members of the population of bit strings will be favored. Bit strings are favored by being replicated more than less favored strings, while less favored strings may be eliminated altogether in a particular generation.

While no false claims are made in this respect, it is also worth noting that the strings in a GA do not self-replicate. They are copied by the simulation system, after evaluation by the fitness function. They may be copied precisely, or with some "mutations" in the form of bit flips, or in many cases, only a portion of the bit string is copied, in combination with a complementary portion from another favored bit string. More flexible GA schemes do not require a fixed length bit string, and in those cases there may also be insertions and deletion of bits in the string.

The GA software totally controls: the form of the solution, the fitness function, the nature of the genetic operators, and the method of replication. It should be noted that GAs do not always conform to the classic form described above. There are many innovations and hybrid forms which can be seen by reviewing any ICGA proceedings, i.e., [2].

2.2 Genetic Programming

In a more recent development, known as genetic programming, the solutions are defined by trees of lisp-like expressions. The genetic operations of mutation and cross-over can operate at any node of the tree. In the case of cross-over, a node is chose at random in (typically) two different trees. The nodes, and all of their higher branches and leaves are simply swapped between the trees. In the case of mutation, the node of one tree could be replaced by a randomly selected node with the same number of descendent branches, leaving the higher branches and leaves unchanged, or a node with more or fewer branches could be put in its place, requiring that some higher branches be added or eliminated. Or, the node and all its higher branches and leaves could be replaced. In the case of mutational branch replacement and addition, the source of origin of the replacement branches and their structure, is arbitrary, and can be generated by a random process.

In this method, the from of the solution does not have to be defined in advance, and so can also evolve. This allows a more creative use of the process of evolution and has been applied to a wide array of problems, notably by John Koza [8, 9].

2.3 Karl Sims – Genetic Images

Karl Sims used the genetic programming method in combination with selection based on the aesthetic criteria of the user to evolve abstract images [17, 18]. This represents a step toward loosening the definition of the "fitness function". In GA and GP, the fitness function is hard-coded into the computational system. In Sims's genetic images, the fitness function is provided by the aesthetically based selections of the human user in each generation of images. These selection criteria are whimsical and change in each generation as the genetic operators generate new arrays of choices.

2.4 Karl Sims – Virtual Creatures

Sims used a similarly flexible genetic system to specify solutions to the problem of being a "creature" built of collections of blocks, linked by flexible joints powered by "muscles" controlled by circuits [19, 20]. Sims embedded these block creatures in a simulation of real physics, such as in water, or on a surface. They were then selected for ability at a variety of tasks, such as swimming, swimming after a light source, moving across a surface, jumping on a surface, and attempting to possess a square block in contest with another creature.

These experiments produced a bewildering and fascinating array of creatures. Some were familiar such as the swimming "snake" and walking "crab" forms. Others were effective at their tasks, but accomplished them with completely unfamiliar patterns of movement and form. This wild proliferation of forms recalls that which occurred among the animals of the Cambrian explosion, where many more experiments were attempted than have survived to be familiar to us today.

In essence, this work is a GP, based on a graph rather than a strict tree structure. Like any GP, the fitness function is predefined (e.g., maximum velocity). However, the objective of the work was not to find an optimal solution, but rather a diversity of solutions for each fitness function. The design of the system was such that a stunning diversity of solutions was possible and did emerge.

An additional loosening of the fitness function took place in the experiment where the objective was to possess a block. Possession of the block involved a competition between two co-evolving populations, tested in individual pairs. Therefore, the form of the competing creature was an evolving part of the fitness landscape. In this case, a large component of the fitness function was outside of the specification of the system. Thus this system exhibits a significant component of natural selection.

2.5 Tierra

My own work with digital evolution follows (conceptually if not chronologically) in this progression of allowing increasing freedom to the evolutionary process. I have set up a system of self-replicating computer programs, where the only clearly defined "fitness" criteria imposed on the system is the same as what is found in organic evolution: the ability to replicate, or transmit genetic material to future generations. My system, called Tierra (Spanish for Earth) creates the basic Darwinian scenario: self-replicating entities in a finite environment with heritable genetic variation, inside the computer [12, 13, 14].

In Tierra, the self-replicating entities are executable machine code programs, which do nothing more than make copies of themselves in the RAM memory of the computer. Thus the machine code becomes an analogue of the nucleic acid based genetic code of organic life. The machine code programs occupy space in the RAM memory, thus the memory provides an analogue of the physical space of organic life.

Genetic variation is introduced into the population of replicating programs by randomly flipping bits in the machine code. This is analogous to mutations involving substitutions of nucleic acids in the DNA sequence of organic life.

The replication of the programs is brought about through their execution by the CPU of the computer. Thus CPU time provides the analogue of the energy that drives the metabolism of organic life.

The results of the evolutions that have been observed so far represent a variety of solutions to the problem of self-replication of machine code programs. We might consider these solutions to fall into two broad classes: "ecological solutions", and "optimizations". Ecological solutions involve interactions between the programs sharing the memory, whereas optimizations involve innovations within the individual algorithms that result in faster replication.

2.5.1 Ecological Evolution

Ecological adaptations arise out of the reality that a predominant feature of the environment is the presence of the digital organisms themselves. Thus digital organisms are able to evolve adaptations to the presence of other organisms, such as parasitism, immunity, cooperation, and cheating. Adaptations have been found for exploiting two valuable resources possessed by digital organisms: CPU time, and information in the form of their genetic (machine) code.

The evolution of ecological adaptations is an ongoing process. It seems that whatever form of algorithm that dominates the memory becomes a target for exploitation (or defense) by other organisms. When the dominant organism becomes the target of a new form of exploitation or defense, it will loose its dominance to the organism possessing the new adaptation, and the cycle begins again.

2.5.2 Optimization

compact code I distinguish "optimizations" from "ecological adaptation" in that optimizations involve innovations within the individual algorithm, not interactions with other individuals. One kind of optimization involves reducing the number of instructions in the algorithm, and the limit of this process appears to lead to a twenty-two byte (non-parasitic) replicator. This replicator is almost a quarter the size of its original eighty byte ancestor, and replicates almost six times as fast.

efficient code However, optimization does not necessarily lead to the most compact code. Some optimizations achieve greater efficiency through more complex code. An example of this involves the technique of "unrolling the loop". The seed program contains a loop which copies one byte from mother to daughter. The program copies its entire genome by executing this loop eighty times. Evolved algorithms have been found to copy two or three bytes per loop, thereby increasing efficiency (calculated by dividing the total number of machine instructions executed in the replication process by the number of bytes in the organism, resulting in a measure of cycles executed per byte copied).

parallel code Another method of speeding the replication process it to parallelize. We have seen this happen opportunistically in the case of "hyper-parasites" which steal CPUs from parasites, and in that way can simultaneously produce more than one daughter.

However, an enhancement was made to the instruction set in which "split" and "join" instructions were added, allowing a process to split into a multi-threaded process, and then optionally to join all the threads back into a single thread. Through evolution, self-replicating algorithms have increased their parallelism to as many as thirty-two processors. Sixteen processor organisms have been found with sixty byte algorithms. Since sixteen does not divide evenly into sixty, the division of the work is more complicated, and the solutions involve some overlap of the data copied by adjacent processors.

3 Evolvability and the Language

A study was conducted comparing the patterns of evolution in four different (but very similar) machine languages [14]. It was found that two of the four showed a much greater magnitude of evolution than the others (measured as optimization through size decrease). Also, the two that showed relatively little evolution, showed a pattern of strict gradualism of evolution, whereas the other two showed abrupt jumps

in evolution (punctuations). Of the two showing punctuations, one demonstrated gradual evolution between the punctuations, while the other showed strict stasis between the punctuations.

It is evident that many aspects of the evolutionary process depend on the structure of the underlying genetic language. Yet, there exists no body of theory relating the structure of the language to its pattern of evolution. This represents a serious hole in our evolutionary theory which was not evident before the advent of synthetic evolution.

This now presents a serious problem for the many engineers who work with evolution as a tool in design or optimization. In every case, the engineer creates a genetic system to describe the solution space, and then evolves that language. Some of these languages will be highly evolvable, while others will not. There is no theory to guide the design of these languages to enhance their evolvability.

4 Evolving the Language

There has been much discussion of how evolution could be allowed to operate on the language of representation itself. One very productive implementation of the idea is that of automatically defined functions [1, 9]. They use Lisp expression trees, in which a special genetic operator is able to convert any sub-tree into a language primitive. These sub-trees are called *modules*, and in essence become frozen tree fragments, which are protected from mutation. As new elements of the underlying language, they become available for insertion by the mutation operator as whole units, and they are protected from fragmentation by the cross-over operator. Therefore they are able to propagate as units through the population.

In this way, the language representation is able to evolve higher level functions. The authors observe: "... the emergence of a useful module reduces the size of the genotypes and, consequently, the number of available crossover and mutation points. This focuses the evolutionary process on improving the evolving programs at a higher level of abstraction rather than destroying previously discovered building blocks." This kind of approach should probably be developed further.

While these methods allow higher level functions to evolve, most discussion of evolution of the representation is focused at the other end: finding a lower level representation, which can allow one to evolve the optimal form of the language representation at its current actual level. I don't know of any examples where this idea has been implemented. While it would be an interesting avenue to explore, I am skeptical of its potential productivity. There has to be a level below which evolution does not operate: the physics. We must ultimately accept that evolution is embedded in a physics, which does not evolve.

5 A Changing Relationship to Evolution

5.1 Complexity Increase

Humans have been practicing applied evolution since the dawn of agriculture, long before the development of the theory of evolution. However, our management of evolution has taken place at the "micro" level, the alteration of existing characteristics of existing species. We have never been able to harness and manage the more creative properties of evolution: the origin of new species, and the emergence of complexity itself. We are able to guide the evolution of poor quality wild corn into high quality domestic corn, however, we can not guide the evolution of algae into corn.

Work with the new synthetic evolutions may allow us to enter into a new relationship with evolution, in which we can manage these more creative aspects of the process. However, this will require new approaches to working with evolution. These higher objectives can not be achieved through the traditional approach of breeding captive populations (applying fitness functions).

5.1.1 Lessons From Nature

Earth's most creative evolutionary transitions were reviewed recently [10]. They seem to occur relatively abruptly, compared to the background rate of evolutionary change. Some of the major transitions noted were: origin of chromosomes, origin of eukaryotes, origin of sex, origin of multi-cellular organisms, origin of social groups.

Of these major transitions, perhaps the most dramatic, and best known, was the rapid origin and diversification of large multi-cellular organisms from micro-scopic single celled ancestors, in what has come to be known as the Cambrian explosion of diversity [4, 11]. It has understandably been called evolution's "big bang", when there was a dramatic inflation of complexity of organisms, and species diversified rapidly into an ecological void.

5.1.2 A Digital Analog to the Cambrian Explosion

Because the Cambrian explosion generated the largest of organic life's complexity increases, it is interesting to consider what its digital analogue may be. At its most fundamental level, the Cambrian explosion arose out of the transition from single to multi-cellular organisms. The digital analog would be a transition from serial to parallel processes.

If we make an analogy between the cell and the processor, then modern multi-cellular organisms are parallel programs on a scale of complexity that vastly exceeds any existing computer software. In organic life, the program is the genome, based on nucleic acid sequences. In humans this program has roughly three billion bases. However, no individual cell expresses all the genes in the genome. Each cell expresses a small subset of the genes, and this subset defines which "cell type" the cell is. It may be a skin cell, liver cell, brain cell, etc. depending on what subset of genes it expresses.

The human body is thought to have several hundred distinct cell types, with a total of trillions of cells. This corresponds to the two main types of parallelism in computer software: SIMD and MIMD. In SIMD parallelism there is a single instruction pointer shared by all of the processors, so every CPU executes the same code in absolute synchrony. In MIMD parallelism, each CPU has its own instruction pointer, so each processor is capable of executing a different set of code.

SIMD parallelism corresponds roughly to multiple cells of a single cell type, in that in both cases, the same genetic code is being expressed. MIMD parallelism corresponds to multiple cells of different cell types in that different cells are expressing different code. Large modern multi-cellular organisms combine both SIMD and MIMD parallelism on a massive scale.

5.1.3 Provoking a Spontaneous Complexity Increase

I believe that evolution can only generate great complexity in the context of an evolving ecological community, as biotic evolutionary interactions are an important driving force in evolution. It appears that this is what happened in organic evolution. In the Amazon region, there are rain forests on white sand soils, where the physical environment consists of clean white sand, air, falling water and sunshine. Embedded in this physical environment is the most complex ecosystem on earth, the tropical rain forest. In this ecosystem there are hundreds of thousands of species. These do not represent hundreds of thousands of adaptations to the physical environment, but rather, most of the adaptations of these species are with respect to the other living organisms that they interact with.

Life transforms the environment, such that the living component of the environment comes to predominate over the physical environment, after which most evolution involves adaptations to other living organisms. Thus the complexity of the living component of the environment comes to greatly exceed the complexity of the physical environment that it is embedded in.

One effort that is underway to create conditions that might provoke evolution to generate substantially greater complexity in digital replicators is what I call the "biodiversity reserve for digital organisms" [15].

This will be based on a networked version of the Tierra software. We will attempt to get thousands of people to run network Tierra on their computers, and all of these computers will be connected into a virtual sub-net of the internet, within which digital organisms will be able to move freely from computer to computer.

The network Tierra software will be run as a low-priority background process, like a screen saver. This means that when the user is actively using the computer, the Tierra software will sleep, receiving no CPU cycles, the energy source for the digital organisms. Any digital organisms who are present on the machine at that time will be frozen, unable to metabolize or reproduce.

This should create a strong selective pressure for individuals to move about the net, avoiding sleeping Tierra programs, and seeking those with a rich supply of CPU cycles. Evolution may generate behaviors that respond to temporal patterns in the availability of CPU cycles. For example, there is likely to be a daily cycle, with generally more free cycles at night when people are sleeping. So some digital organisms might evolve the behavior of migrating around the planet on a daily basis staying on the dark side of the planet.

It is hoped that the more complex and dynamic environment presented by the network will challenge evolution with a more complex problem than the basic string copy of the single node installation. This should provide selective forces to initiate an increase in complexity. Once a significant impulse in the direction of complexity has occurred, the hope is that selective forces arising from interactions among the digital organisms can lead to an auto-catalytic increase in complexity.

The first evolution observed in Tierra was the origin of ecological interactions, which were based on adaptation to the presence of other digital organisms in the environment. Thus this dynamic has been present in Tierra from the beginning. It is hoped that with the help of some impulse towards greater complexity, this dynamic can lead to a large spiraling upwards in complexity.

5.2 Returning to our Roots

How might we work with digital evolution to produce useful products? Although we have a well established practice of plant and animal breeding, our relationship to digital evolution is quite different. Our ancestors were able to go out into nature and observe many highly evolved and complex organisms. They found uses for some of these, such as the ancestors of, rice, corn, wheat, chickens, pigs, dogs, etc. They then bred them to produce the much improved domesticated plants and animals that we know today.

However, in the case of digital evolution, we are starting with very simple organisms that have not yet achieved the complexity to be useful, so our first objective is to evolve complexity. If a digital analog to the Cambrian explosion can be achieved, then it should be possible to return to the same kind of relationship to digital evolution that we have with organic evolution. We can go out into digital nature, and observe the complex products of evolution. We can observe them for interesting and potentially useful information processes. When we identify potentially useful digital organisms, we can capture them and subject them to selective breeding to enhance their performance on the application, and inhibit unruly wild behavior. Eventually the product can be neutered and sold to the end user.

We have seen the tremendous creative potential of evolution when expressed through the medium of organic chemistry. We do not yet know the full potential of evolution in the medium of digital computation. However, the initial experiments have been very promising, suggesting that it is worthwhile to make the effort to push digital evolution to its limits. If digital evolution has even a small fraction of the potential of organic evolution, it could result in information process of a complexity far beyond anything that we have experience with today. While there are many potential obstacles and technical problems along the way, the possible rewards for success make the risk worth taking. Yet it is a venture into the unknown for which we can not estimate the likelihood of success.

References

[1] Angeline, Peter J., and Jordan B. Pollack. 1994. Coevolving high-level representations. *In*: Artificial Life III, [Ed.] Christopher G. Langton, SFI Studies in the Sciences of Complexity, Proc. Vol. XVII, Addison-Wesley. Pp. 55–71.

[2] Belew, R. K., and L. B. Booker [eds.]. 1991. Proceedings of the 1991 International Conference on Genetic Algorithms, Pp. 576. San Mateo, CA: Morgan Kaufmann.

[3] Goldberg, D. E. 1989. Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley.

[4] Gould, Steven J. 1989. Wonderful life. W. W. Norton & Company, Inc. Pp. 347.

[5] Holland, John Henry. 1975. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence (Univ. of Michigan Press, Ann Arbor).

[6] Hong, J. I., Feng, Q., Rotello, V. & Rebek, J. 1992. Competition, cooperation, and mutation: improving a synthetic replicator by light irradiation. Science 255: 848–850.

[7] Joyce, Gerald F. 1992. Directed molecular evolution. Scientific American, December 1992: 90–97.

[8] Koza, John R. 1992. Genetic programming, on the programming of computers by means of natural selection. Cambridge, MA: MIT Press.

[9] ———. 1994. Genetic programming II, automatic discovery of reusable programs. Cambridge, MA: MIT Press.

[10] Maynard Smith, J. and E. Szathmáry, The Major Transitions in Evolution (Oxford: Freeman, 1995).

[11] Morris, S. Conway. 1989. Burgess shale faunas and the Cambrian explosion. Science 246: 339–346.

[12] Ray, T. S. 1991. An approach to the synthesis of life. *In*: Langton, C., C. Taylor, J. D. Farmer, & S. Rasmussen [eds], Artificial Life II, Santa Fe Institute Studies in the Sciences of Complexity, vol. X, 371–408. Redwood City, CA: Addison-Wesley.

[13] ———. 1994. An evolutionary approach to synthetic biology: Zen and the art of creating life. Artificial Life 1(1/2): 195–226. Reprinted *In*: Langton, C. G. [ed.], Artificial Life, an overview. The MIT Press, 1995.

[14] ———. 1994. Evolution, complexity, entropy, and artificial reality. Physica D 75: 239–263.

[15] ———. 1995. A proposal to create a network-wide biodiversity reserve for digital organisms. ATR Technical Report TR-H-133. Available at:
http://www.hip.atr.co.jp/~ray/pubs/reserves/reserves.html

[16] Sanchez, E., and M. Tomassini [eds.]. 1996. Towards evolvable hardware, Lecture notes in computer science, Vol. 1062, Springer-Verlag, ISBN 3-540-61093-6

[17] Sims, K. 1991. "Artificial Evolution for Computer Graphics," Computer Graphics (Siggraph '91 proceedings), Vol.25, No.4, July 1991, pp.319-328.

[18] ———. 1993. "Interactive Evolution of Equations for Procedural Models," The Visual Computer, Vol.9, No.8, August 1993, pp.466-476.

[19] ———. 1994. "Evolving Virtual Creatures," Computer Graphics (Siggraph '94) Annual Conference Proceedings, July 1994, pp.15-22.

[20] ———. 1994. "Evolving 3D Morphology and Behavior by Competition," Artificial Life IV Proceedings, R. Brooks and P.Maes [eds.], MIT Press, 1994, pp.28-39.