Turing's Dreampond

Gordon Briggs
gmb35@cornell.edu

December 3, 2007

1 Introduction

> One of the phenomena which had peculiarly attracted my attention was the structure of the human frame, and, indeed, any animal endued with life. Whence, I often asked myself, did the principle of life proceed? It was a bold question, and one which has ever been considered as a mystery...When I found so astonishing a power placed within my hands, I hesitated a long time concerning the manner in which I should employ it. Although I possessed the capacity of bestowing animation, yet to prepare a frame for the reception of it, with all its intricacies of fibres, muscles, and veins, still remained a work of inconceivable difficulty and labour. I doubted at first whether I should attempt the creation of a being like myself, or one of simpler organisation; but my imagination was too much exalted by my first success to permit me to doubt of my ability to give life to an animal as complex and wonderful as man.
>
> - Mary Shelley, *Frankenstein*

As Shelley's tragic figure of Victor Frankenstein was consumed by his pursuit to manufacture life, so has the fascination with the genesis of life and intelligence dwelled within humanity's collective consciousness. We have extensively pondered not only our own origins, but the possibility of playing the role of life's originator. The Greeks narrated how Pygmalion sculpted Galatea. In Ovid's Metamorphoses, Prometheus' sin was not of giving man fire, but of imbuing man with life, having sculpted him from the dust of the earth. Indeed, modern science fiction now abounds with cautionary tales of our intelligent creations running amok and usurping our dominance on Earth.

The potential to create machines that act as living, thinking beings do inspires and evokes passionate opinions within ourselves. It raises fundamental questions regarding the

cognitive nature of our creations and our own sapience. Can machines think? How do we tell if an entity is truly sentient? Are we ultimately machines ourselves? How do we create machines that think? Philosophers and scientists have contemplated and debated the answers to these inquiries for years, while authors and storytellers have woven countless stories exploring their implications.

1.1 Artificial Intelligence

However, unlike the literary and mythic characters who sought to build artificial beings through alchemical processes and petitions to the divine, the modern field of Artificial Intelligence (AI), which concerns the understanding and building of intelligent systems [3], involves the application of mathematical and computational theory in order to generate rational solutions to complex problems.

Modern AI can trace its own origins back to around the early 1950s, roughly corresponding to the advent of the modern computer [2]. AI's early pioneers programmed computers and devised algorithms that tackled problems ranging from game-playing to theorem-proving. Allen Newell and Herbert Simon, developed the Logic Theorist, a program capable of proving theorems in logic, which they applied to various theorems in the work Principia Mathematica [3]. Arthur Samuel of IBM authored a checkers-playing program that eventually reached world-class player level [2]. In 1965, Joseph Weizenbaum of MIT constructed ELIZA, a program that conversed with a human user in English [2].

These are but a few of the instances of programs that were developed in the early days of AI that successfully performed complex and intelligent actions. They utilized techniques based on of formal symbol manipulation, such as comprehensive or guided searches of a problem's state space and logical inference and resolution algorithms. Even the task of emulating human conversation, as ELIZA could, was accomplished using such traditional AI approaches. These traditional AI approaches, involving the application of logical and mathematical principles, constitutes a top-down engineering process, in which the designer is fully cognizant of all the dynamics involved with the environment and designs the intelligent system accordingly.

1.2 Evolution and AI

Even though the field of AI is often seen as the endeavour of man as an intelligent designer, there has always been strong connection between the fields of evolution and AI. Alan Turing, one of the founders of the field of Computer Science and AI, wrote a seminal article in 1950, entitled "Computing Machinery and Intelligence", in which he presented his famous Turing test and discussed the prospects of developing intelligent machines. However, in this article, Turing wrote [1]

> There is an obvious connection between this process and evolution, by the identifications: Structure of the child machine = Hereditary material; Changes = Mutations; Natural selection = Judgement of the experimenter
>
> One may hope, however, that this process will be more expeditious than evolution. The survival of the fittest is a slow method for measuring advantages. The experimenter, by exercise of intelligence, should be able to speed it up. Equally important is the fact that he is not restricted to random mutations. If he can trace a cause for some weakness he can probably think of the kind of mutation which will improve it.

Turing's description of the human creative process, however accurate, is perhaps excessively optimistic when applied to our desires to construct intelligent agents. Our ability to ascertain what changes of the machine must be made in order to create a machine with higher cognitive abilities by the exercise of [our] intelligence is woefully slim. The fact is, our knowledge of our human cognition and the ability to characterize and explain it is still tremendously lacking. Yet, it is quite fascinating that Turing chooses to compare our quest to create intelligent machines to the evolutionary process.

Concerning the genesis of life on Earth, the esteemed evolutionary biologist Stephen Jay Gould wrote in Time magazine [4]:

> The life that we know, however wondrous in extent and variety, all proceedsor so our best inferences tell usfrom one single experiment. The biochemical features underlying this amazing variety and the coherent fossil record of 3.5 billion years (implying a single branching tree of earthly life with a common trunk) indicate that every living thing on Earth, from the tiniest bacterium on the ocean floor to the highest albatross that ever flew in the sky, arose as the magnificently diversified evolutionary outcome of one single experiment performed by nature, one origin of life in the early history of one particular planet

Intelligent life and the cognitive abilities found manifest in it are emergent outcomes of evolution. Since the evolutionary process has already generated instances of sentience, we cannot ignore this mechanism in our own attempts to build intelligent systems. I wish to explore this topic by discussing a few biologically-inspired techniques found in the field of AI, applying them to a simple problem, and then discussing the vast implications and avenues of further study that logically follow.

2 Biology and AI

Computer Science, especially in the field of AI, has drawn much inspiration from emulating nature. There are two notable techniques that seek to model phenomena found in the biological world, Artificial Neural Networks and Genetic Algorithms.

2.1 Artifical Neural Networks

Artificial Neural Networks (ANNs) were first implemented in 1951 by Marvin Minsky and Dean Edmonds, building the SNARC neural network computer based off of the mathematical model of neurons developed by Warren McCulloch and Walter Pitts [3]. However, there was a lull in neural network research for a few decades until the advent of the backpropagation algorithm for supervised network training and its successful implementation (around the early 1980s). ANNs attempt to emulate the operation of a brain my modeling it at the neuronal levelmodeling complex networks of interconnected individual neurons.

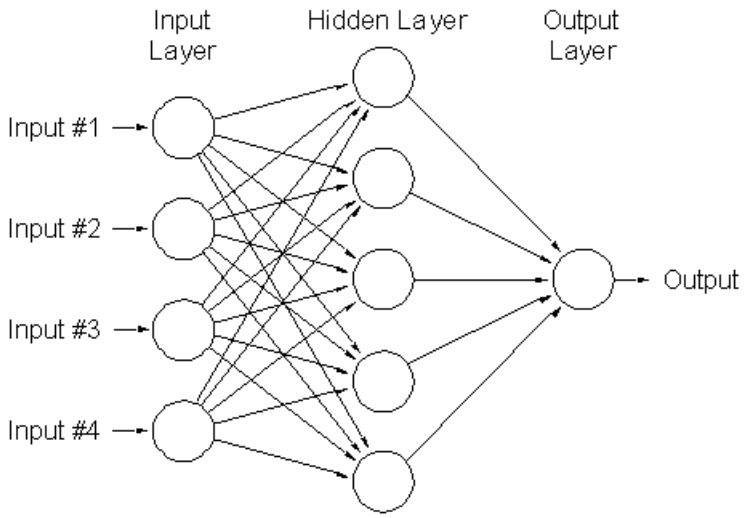

Figure 1 - Model of an ANN [5] .

Each neuron is modeled with several input links, each assigned a bias weight, and some number of output linksusually connected to the input links of other neurons. The bias weights act as approximations to the inhibitory and excitatory connection behaviors found in actual neurological structures. In order to generate an output signal, the neuron sums the product of each input signal and the associated bias weight, then applies some nonlinear activation function to this sum [3].

The strength of ANNs lies in two attributes. First, as a purely mathematical construct, ANNs are able to approximate any continuous function given the appropriate network topology. This result was demonstrated by the Cybenko Theorem [7]. The second strength of ANNs involve the abilities of these networks and simple units to learn and adapt to complex tasks such as pattern recognition. Learning is simply a matter of appropriately adjusting each bias weight to generate better output values, either be comparing the outputs to desired outputs, as in supervised learning techniques such as backpropagation, or through some unsupervised learning such as Hebbian learning. Connecting these artificial neurons in increasingly complex network topologies yields increasingly complex system behavior. Recurrent neural networks (RNNs), which contain backward neuronal (feedback) connections, have been demonstrated to be quite dynamic and hard to characterize. Certain RNN

configurations have even demonstrated abilities akin to short-term and long-term memory. These dynamic and hard to characterize behaviors can be regarded as emergent properties of the ANN architecture.

2.2 Genetic Algorithms

Evolutionary algorithms, also known as genetic algorithms (GAs) are stochastic, heuristic search algorithms inspired by natural evolutionary processes. The general scheme of a GA involves a population of randomly generated individuals, represented by a set of genes (implemented with binary bit strings or some other method of trait encoding) each describing a phenotypical expression. For instance, if we wanted to use a binary encoding for the length of a limb, we could use the following scheme.

$$g(i) = x_i^1 x_i^2 x_i^3 ... x_i^n$$

where $g(i)$ is the genotype of individual i, and x_i^j is a 'gene' with value either equal to 0 or 1. Thus the phenotypic expression for limb length could be derived from the genotype by:

$$p(i) = \sum_{j=1}^{n} x_i^j$$

Thus we see if we had an individual i such that $g(i) = 0100011$, we would see that $p(i) = 0 + 1 + 0 + 0 + 0 + 1 + 1 = 3$. Individual i would have a phenotypic expression of 3, whatever that means in the given context (limb length, etc.).

Every individual in the population is evaluated and assigned a fitness value by some fitness function. Subsequently, a new generation is created by randomly selecting individuals, favoring those with higher fitness, to breed. This breeding involves the exchange of genotype data through a probabilistically governed process called crossover, the frequency of which is determined by a crossover rate. If crossover breeding occurs, then genetic information is swapped through some crossover operation, else the offspring individuals will be identical to parent individuals. Mutation of offspring gene data can also occur, the frequency of which

is determined by a mutation rate and is computed during each breeding. For instance, using our above binary encoding scheme, the creation of a new individual might look something like:

$$g(1) = 0110011$$
$$g(2) = 1001000$$
$$Child(g(1), g(2)) = 10|10011 = 1010011$$

As one can see, the crossover operation used above was simply taking the genetic material from one individual at some point and combining it with the other at the corresponding point.

This procedure is repeated for a set number of generations, or until some fitness criterion has been fulfilled [3]. Essentially, genetic algorithms implement a artificial selection process, weeding out individuals ranked poorly by an arbitrarily defined fitness function and conversely propagating the genotypes of individuals that do satisfy the fitness function. The mutation and crossover procedures are designed to generate new individual designs to explore all possible solutions to the optimization problem posed by the fitness function. Mutation would constitute a local search of all individual solutions near a given individual, while crossovers would comprise a more global heuristic search, guided by the traits of parents.

Natural evolution occurs in similar algorithmic framework, though it is clear that GAs are still quite high-order approximations of what occurs in nature. The complex biochemical operations found in DNA recombination and mutation are approximated using probabilistically determined crossover and mutation operators. The natural selection of fit individuals, manifested in the death of unfit individuals and relative prosperity of fit individuals in nature, has been again approximated as a stochastic operation. Genetic algorithms rely on the emergent phenomenon of the propagation of better designed individuals throughout the simulation generations in order to properly function.

3 Pole Position

In order to demostrate both the aforementioned biologically inspired AI techniques, we will apply both methods toward solving a basic control problem. The problem of pole balancing is often used to develop and test control systems. Clearly, this endeavour is not at the level of beating Gary Kasporov at chess, but it does serve to illustrate how we can apply these techniques to develop intelligent behaviors. The problem involves maintaining a vertically oriented pole within some angle of the vertical. The difficult lies in the fact that the pole is precariously balanced on a cart that must constantly be in motion. There is a constant force acting on the cart that can either be directed to the left or right. Therefore, it is the task of the control algorithm to define a function f such that

$$F = |F|\text{sign}(f(\theta, v)) \tag{1}$$

that is to say when the function f is negative based of the pole's angle θ and velocity v , the force is directed to the left (in the negative direction), and if f is positive the force F is directed to the right. An intelligent control algorithm should be able to maintain the pole's balance. So how do we go about developing such a control algorithm? One might first be tempted to try to develop one him or herself. Surely, this sounds like a physics problem that someone with a mathematical inclination could reason out. Let us then examine the physical model that will simulate the problem.

3.1 Physical Model

In order to model the movement of the pole, I used a model described in [6]. We must keep track of a multitude of variables that I have listed below:

Variable	Description
r	Length of the Pole (in meters)
g	Force of gravity (-9.81 $\frac{m}{s^2}$)
m_p	Mass of the Pole (in kg)
m_c	Mass of the Cart (in kg)
F_t	Force acting on the cart at time t. (Remember could be either $-\|F_t\|$ or $\|F_t\|$).
δ	Simulation timestep (in seconds)
θ_t	Angle of the Pole at time t
$\theta_t^{'}$	Angular velocity of the Pole at time t
$\theta_t^{''}$	Angular acceleration of the Pole at time t
x_t	Position of the Pole/Cart at time t
$x_t^{'}$	Velocity of the Pole/Cart at time t
$x_t^{''}$	Acceleration of the Pole/Cart at time t
θ_{max}	Maximum angle the Pole is allowed to deviate from the vertical.

The equations of motions provided by [6] are

$$\theta_t^{''} = \frac{g\ sin(\theta_t) + cos(\theta_t)\frac{-F_t - m_p r \theta_t^{'2} sin(\theta_t)}{m_c + m_p}}{r(\frac{4}{3} - \frac{m_p cos^2(\theta_t)}{m_c + m_p})} \tag{2}$$

$$x_t^{''} = \frac{F_t + m_p r(\theta_t^{'2} sin(\theta_t) - \theta_t^{''} cos(\theta_t))}{m_c + m_p} \tag{3}$$

The updates in the model are simulated using Euler's method. Thus we obtain:

$$x_{t+1} = x_t + \delta x_t^{'}$$
$$x_{t+1}^{'} = x_t^{'} + \delta x_t^{''}$$

$$\theta_{t+1} = \theta_t + \delta \theta_t^{'}$$
$$\theta_{t+1}^{'} = \theta_t^{'} + \delta \theta_t^{''}$$

Clearly, this is a rather involved system. The equations of motion are sufficient complex (and ugly) such that it would be rather painful to attempt to derive our desired control function $f(\theta, v)$. So instead, let us attempt to evolve a solution.

3.2 Our Solution

We know that ANNs can approximate any continuous function given the appropriately chosen weights. We are going to use a genetic algorithm to choose those weights for us.

3.2.1 ANN

Since we are attempting to develop a control function $f(\theta, v)$, we need a neural network with two inputs and one output node. We are developing a feedforward network with one hidden layer (much like the one in figure 1, except that it only has two inputs instead of the four picture). We will use the hyperbolic tangent function as our non-linear activation function. If we let N be the number of hidden neurons in our network then we can see we need two vectors of weights W_h and W_o. W_h are the weights that affect the inputs to the hidden layer and W_o are the weights that affect the inputs to the output node. Since there are two input nodes that must be feed into N hidden nodes, we see that $|W_h| = 2N$. Since there are N hidden neurons that feed into just one output node, we see that $|W_o| = N$. Our control function is thus expressed

$$f(\theta, v) = tanh(\sum_{k=1}^{N} w_o^k \cdot (tanh(\sum_{j=1}^{N} w_h^{2j}\theta + w_h^{2j+1}v))) \tag{4}$$

where w_o^k is the k-th weight in vector W_o and w_h^j is the j-th weight in vector W_h.

3.2.2 GA

Let us consider an individual solution to be the pair $W = (W_h, W_o)$. We will construct a population of size P that is comprised of P invidiuals. We initialize the first generation with random weights in the domain of [-1,1]. In order to evalute the fitness of an individual solution, we use the control function generated by its weights W to control the pole. We then

obtain a value $fitness(W)$, where $fitness(W)$ is the number of timesteps of the simulation the pole's angle remained within $(-\theta_{max}, \theta_{max})$. We then use these fitness values to select individuals to help produce the new generation. In order to produce an individual in the new population, we randomly select two individuals from the old population. The probably that an individual will be selected is from a population P is:

$$S(W) = \frac{fitness(W)}{\sum_{X \in P} fitness(X)} \tag{5}$$

that is we normalize the probability of W begin chosen with the total fitness score of the entire population. When two individuals are selected, we produce a child solution using mutation and crossover operations on their weights. We repeat this until the new generation has been filled.

We implement mutation and crossover in the following manner:

- Mutation : We assign a mutation rate μ. Such that when we are generating a new individual, each weight in the new ANN has a μ chance of being assigned a random value (in the domain [-1,1]).
- Crossover : If mutual on a given weight does not occur, we generate a new weight using a crossover operation. We generate a random weight $\sigma \in (0, 1)$ such that the weight for the new individual is $w_{new} = \frac{\sigma w_a + (1-\sigma) w_b}{2}$, where w_a is the weight for individual a and w_b for individual b. So, we basically take a randomly weighted average between the two weights.

This process is repeated for some arbitrary number of generations.

3.3 Implementation

I wrote the code to perform the simulation in C code and used OpenGL's GLUT library to render the graphics. It was compiled and run in a Linux environment. I have included a screenshot below.

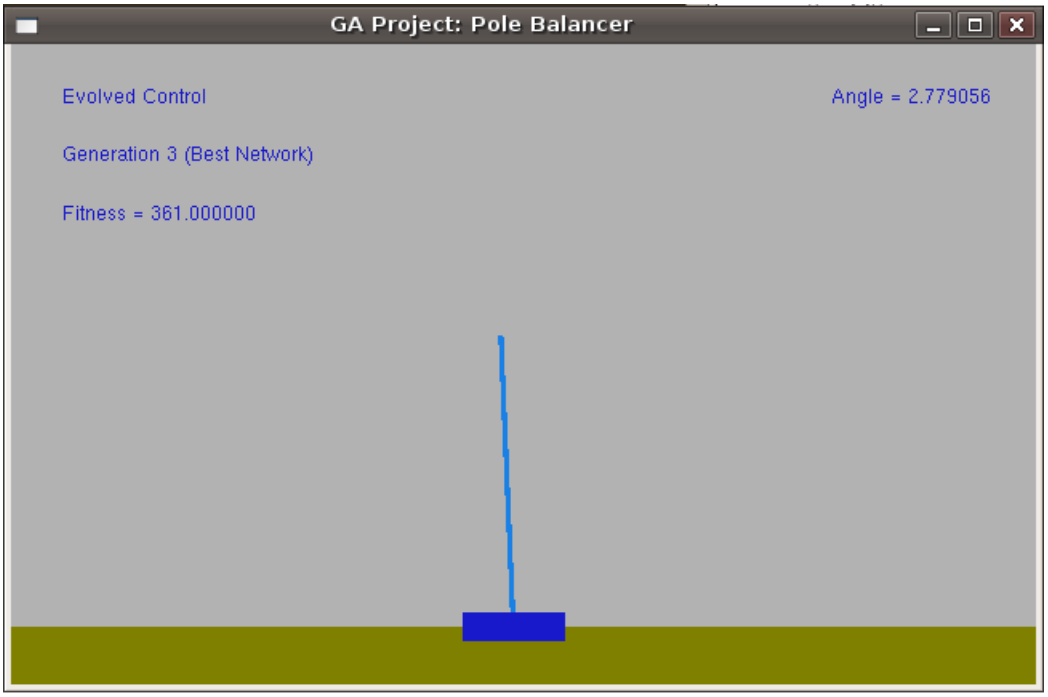

Figure 3 - Screenshot of the GA Program

3.4 Results

I proceeded to run 20 trials (each with a new randomly generated initial population), evolving the populations for 50 generations. I used a timestep of $\delta = 0.002$ and a $\theta_{max} = 12.5°$. The pole was initialized with a near zero angle, $m_p = 0.1$, $m_c = 1$, $r = 5$, and both the pole and cart were at rest. I used a neural network with 6 hidden units. I averaged the average fitness value at each generation for each trial. The results are found below:

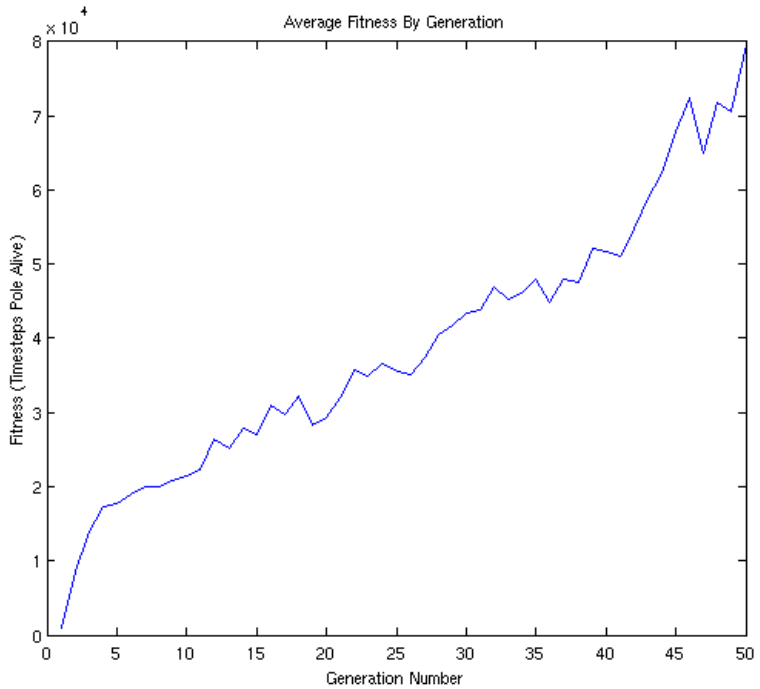

Figure 4 - Results with Crossover

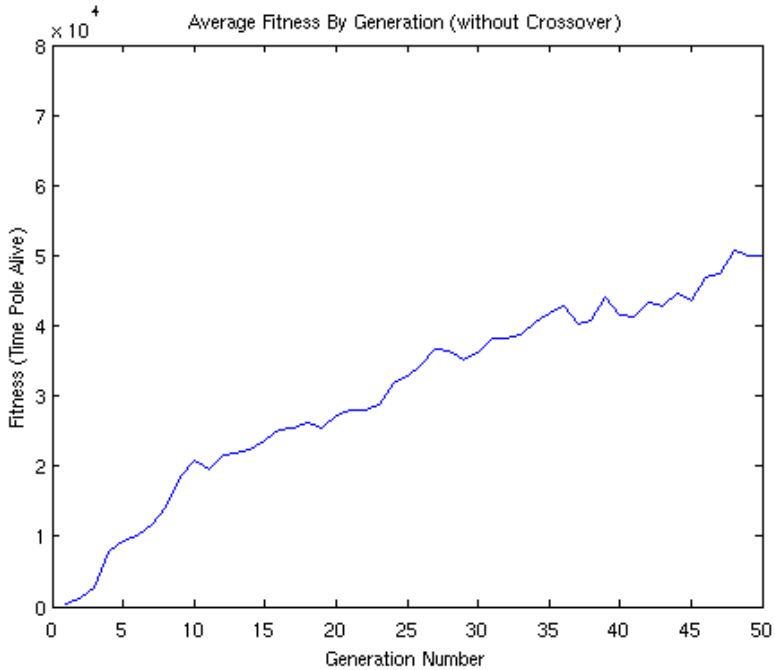

Figure 5 - Results without Crossover

We can see that evolution with crossover performs better than evolution without crossover, although both types did succeed in generating good control functions.

3.5 Conclusion

We have demonstrated that decent solutions to the pole-balancing problem can be easily evolved using a genetic algorithm. This feat was accomplished without having any *a priori* notion of how the control function should work. The elegance of genetic algorithms is that you simply need a way to express all your possible solutions and evaluate how fit they are. As load as you keep favoring the propogation of the more fit solutions and performing random mutations and crossover to generate new solutions, you will implicitly solve the problem! We did not need to perform extensive mathematical characterization of the equations of motion found above; no intelligent design is necessary.

In the following section, we will discuss the possibilities that arise out of the application of evolutionary computation.

4 Endless Forms

The field of Artificial Life (AL) seeks to evolve unique simulated lifeforms, seeking to develop interesting physical and behavioral adaptations. Virtual environments are created, complete with a basic set of rules that govern them (often simulated physics). AL applies GA techniques to evolve simulated creatures that are optimized for these environments.

4.1 Artificial Life

One of the prominent individuals in the artificial life movement is Karl Sims, who generated a virtual environment in an effort to evolutionarily optimize the design and motions of artificial creatures that swam in water [8]. Sims simulated fluid mechanics in order to govern the motion of the simulated creatures, which were composed of a series of jointed boxes. He also used ANNs to govern the behavior of these simulated creatures providing the appropriate swimming behaviors, or perhaps in the worst cases: futile spastic motions. The fitness

function used was maximum velocity, so those who could swim faster were evolutionarily favored. Sims' simulations were successful in generating a wide variety of unique solutions to the problem of swimming. For instance, some of the evolved individuals resembled water snakes or crabs, whereas others generated totally unfamiliar body structures and swimming behaviors, but were still nonetheless effective swimmers [8]. Numerous other studies have been conducted applying similar techniques to other virtual scenarios, e.g. Walking on land, predator-prey models.

Figure 6 - A example of Karl Sim's swimming snakes[1].

4.2 A New Frontier

Having discussed the efficacies of natural evolution and evolutionary computation, we are left with an exciting possibility involving our abilities to simulate and approximate the natural world. How can we characterize the natural world? Our physical existence is governed by a complex set of physical laws (gravity, electromagnetism, etc). Our physical universe is comprised of various subatomic particles, the material that these physical laws operate upon. Changes in the universe are manifested by a progression though a temporal dimension, time. The interactions of these physical laws, the material building blocks, and time constitute a causal architecture that gives rise to all the physical phenomena in our uni-

[1] http://www.genarts.com/karl/evolved-virtual-creatures.html

verse. That eventually some self-organizing, self-replicating structures arose, giving birth to primitive life. These self-organizing structures and physical phenomena produce by these lower level emergent properties establish a causal architecture that yields the evolutionary process. This process of self-replication was governed by the evolutionary process, and this evolutionary process eventually generated humans and human cognition. This characterization of the natural world constitutes something we can pedantically deign an Existential Paradigm. Therefore, a set of simulated physics, simulated building blocks, and a simulated progression through time would constitute a Existential Simulacrum Paradigm. We can classify such as Karl Sims swimming creatures experiment, as an implementation of an existential simulacrum paradigm. He utilized a set of simulated physical laws (fluid dynamics), with a set of basic building structures (boxes), and of course simulated the system through time. Yet, we can see that this existential simulacrum paradigm is tremendously simple compared to the actual existential paradigm. What would happen if we implemented something more complex?

Thus I propose the following Gedankenexperiment. Assume that you had an effectively unlimited about of computation power at their disposal. You would then be enabled to closely simulate the complex interactions that occur in nature. Therefore, the question arises, when our Existential Simulacrum Paradigm, implemented on our powerful computation system approaches the same level of detail as the actual Existential Paradigm, would the systems that emerge through the resulting evolutionary processes be correspondingly complex? Consequently, would the intelligent simulated lifeforms, be corresponding more life-like? Would human-like beings eventually emerge? Would their brains, be anything like a human brain? Could we use this simulation, if not a tool to create artificial life itself, as a blueprint to engineer it in the real-world? Certainly, such a fantastical possibility succeeds in raising innumerable questions such as those aforementioned. I would be inclined to say yes to all those inquiries, given what we know about our own existence, and the hypothetical similarity of the simulated existence we create. Yet, the genesis of artificial intelligence and life in an Existential Simulacrum Paradigm, like all emergent phenomena, is to be unexpected. In order to allay our curiosity, we must actively implement and study such a simulated

environment. Unfortunately, we do not have enough computational resources to implement this Existential Simulacrum Paradigm. We cannot even build high-fidelity simulations of meteorological processes to predict the weather, how can be expect to model the cosmological workings of the universe from the quark up? Indeed, even humanity's understanding of the laws of the physical universe is also an ever-evolving and improving endeavor. Perhaps, then, it is a foolish activity to muse over such a hypothetical and far-distant possibility. Such a comprehensive and holistic simulation of the physical world is certainly not possible in the near-future. Yet, despite this pessimistic outlook, we may acknowledge that work in studying Existential Simulacrum Paradigms (e.g. Karl Sim's water snakes) is ongoing ventureone that is yielding fascinating results. Knowing, then, what has been accomplished, what lies ahead, and how far we have to go is perhaps is the most tantalizing prospect in evolutionary computation.

References

[1] Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind*, Vol. LIX, No. 236, 433-460.

[2] Buchanan, Bruce G. (2000). Brief History of Artificial Intelligence. http://www.aaai.org/aitopics/bbhist.html

[3] Russell, Stuart and Norvig, Peter (2003). Artificial Intelligence: A Modern Approach. Upper Saddle River, NJ: Prentice Hall.

[4] Gould, Stephen Jay. "Will We Figure Out How Life Began?" April 10, 2000; Time magazine, 155 (14): 92-93.

[5] Rounds, Stewart A. (2002). "Development of a Neural Network Model for Dissolved Oxygen in the Tualatin River, Oregon." http://smig.usgs.gov/SMIG/features_0902/tualatin_ann.html

[6] Brownlee, Jason. (2005). The Pole Balancing Problem: A Benchmark Control Theory Problem. http://www.ict.swin.edu.au/personal/jbrownlee/2005/TR07-2005.pdf

[7] Mitchel, Tom (1997). Machine Learning. Boston, MA: McGraw-Hill.

[8] Ray, Thomas. (1996). Karl Sims- Water Snakes. http://www.his.atr.jp/ ray/pubs/fatm/node10.html