
A Study of Evolutionary Acceleration

S. Kazadi, S. Cheung, C. Ogletree, S. Kim, C. Lee, A. Min

Jisan Research Institute
28 North Oak Avenue
Pasadena, CA 91107
USA

Abstract

We investigate the phenomenon of numerical evolutionary acceleration. This phenomenon is a simple consequence of numerical analysis of the probabilities of evolving independent parts of a complex system in the presence of evolutionary epochs. The epoch mechanism allows the newly evolved structure to become part of the overall system design of all elements of the population. We demonstrated that this phenomenon not only exists in real evolving systems, but that evolutionary acceleration dwarfs the group mechanism for some complex structures.

keywords: evolutionary design, numerical evolutionary acceleration

1 Introduction

In studying evolving systems, we have observed some interesting and thus far (as far as these authors are aware) undocumented effects of the evolutionary process. We have understood these effects in terms of a rather simple simulation that was originally intended to explore the development of compartments¹ in evolutionary systems. The development of these compartments seemed to be a natural result of the evolutionary process, as expected. However, we noticed that the evolutionary effect of compartmentalization seemed to differ widely depending on the way in which the compartments were put together, and the way in which the individual evolutionary pieces interact with each

¹A piece of DNA that defines a fitness-active compartment [Kazadi, 2000], or a compartment which controls the design of part of the design which has an impact on the designs fitness.

other. Moreover, the evolutionary time did not seem to increase significantly when compartments were not used, contrary to our expectations.

This apparent contradiction of common sense was very intriguing. Why would the evolutionary time not increase factorially when the complexity of devices increased, making compartments an increasingly important part of the evolutionary process? Answering this question led to an investigation of a phenomenon we now call *numerical evolutionary acceleration* (NEA) which is the topic of this paper. This phenomenon, inherent in nearly all evolutionary systems, is a population-based phenomenon which emerges only in large evolutionary systems. The overall effect is the dramatic increase in the evolutionary speed of a particular system, yielding an evolutionary time that approaches a linear relationship with the sum of individual steps' evolutionary times. Once understood, NEA may become an important part of artificial evolutionary design.

This paper takes a first look at NEA. We explore the evolution of a number of related devices, the evolution of which may be used to probe the effectiveness of NEA. The remainder of the paper is organized as follows: Section 2 describes the evolutionary acceleration phenomenon and Section 3 describes our simulation and data we collected and analyzed from our simulation.

2 Evolutionary Acceleration

Evolutionary acceleration is a natural phenomenon whereby the evolution of a population speeds up through a series of epochs. Often times, within individuals in a population, advantageous features will arise. Over several iterations of the population's development, other individuals without the improvement will acquire the advantageous trait of that one improve

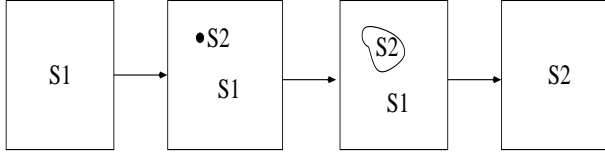


Figure 1: Schematics of an epoch.

individual. This development of the population is referred to as an epoch.

Consider a population with an initial state of S_1 . In this population, all the individuals start with a fitness score of $x, x \in R$. Through random mutations, an individual with a fitness score of $x + \Delta x, \Delta x \in R$ appears. Because this individual's structure is closer to the desired state, it is used to propagate through the population, eliminating the surrounding individuals with a fitness score of x . Essentially this process is the epoch that occurs within the population. We will see that a series of these epochs cause the speed of the evolution occurring in the population to increase, thus evolutionary acceleration appears.

This process repeats until the population reaches S_n , where n is the desired state of an individual.

Each individual, N , that is a part of the population has a probability, p_l where $1 \leq l \leq N$, of developing a beneficial trait which helps in the individual's survival. If the probability is small, and many individuals are in the population, one might expect that the time for evolution of a single unit will be given by

$$\tau = k \frac{1}{p_l} \quad (1)$$

This time is for a single change instead of multiple changes. For multiple changes in the same individual, the time expected to find the appropriate evolutionary step (a new epoch) is

$$\tau = k \frac{1}{\prod_i p_i}$$

The result would be the time required before we could expect to find a single individual exhibiting all of the characteristics in question. The number of individuals required in order to generate a single one would also be of the same order as the time τ . As an example, if there were five different characteristics, each with a probability of occurrence of 0.1, then the number of individuals would be 10^5 . The time taken for the complete evolution of a device is the inverse of the *product* of probabilities.

If epochs occur after each improvement, however, the time an individual will take to evolve all improvements will be the inverse of the *summation* of probabilities of the historical steps taken up to the end. The time to completion of the task would simply be the sum of times required for all different parts to be independently found. That is, the time would change over to

$$\tau = \sum_l \frac{1}{p_l}$$

In the previous example, the value would be 50.

3 Computational Trials

In this section, we will discuss our evolutionary algorithm and the data produced by the simulation.

3.1 Simulation

We utilize a simulation which consists of population of evolving individuals that live in a toroidal lattice. Each individual contains a set of genetic information, or DNA, and a fitness score. The DNA specifies the structure and function of the network. The fitness score is a measure of the individuals' closeness (in functionality) to the desired device.

Each individual in the lattice is a network of nodes and connections. We use three different types of nodes in our networks: input, firing, and output nodes. The input nodes are signal receptors and are the networks only means of acquiring external data. We use integer data values as inputs to the nodes. The firing nodes are transceivers of data. Each one is a lookup table which contains the output number selected by the input number. For instance, considering a firing node with an IO table of 1200111111, the input number of 0 will choose the first element in the IO table and output that number, which in this case is a 1, whereas an input value of three will output the fourth element of the IO table, which in this case is a 0. Output nodes are transmitters of data values, and represent the way in which data goes from the network to the external environment.

These various types of nodes are interconnected with connections. These connections flow in the general direction of input nodes to output nodes, with firing nodes forming the middle bulk of the network. Because there are no restrictions in having multiple connections between the nodes, it is possible to generate connections identical to already existing connections. Several of these repeats will multiply the number of

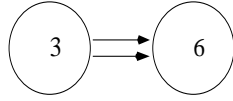


Figure 2: Nodes connected by multiple connections.

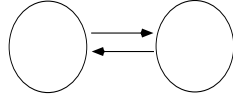


Figure 3: Nodes connected by connections going in the opposite directions.

outputs sent. For example, twice the original information will be sent if there are two connections between the nodes going in the same direction, as in Figure 2.

One might also find two nodes with connections heading in opposite directions. Between two nodes A and B, a connection with a starting point of node A and a ending of node B will be complimented by another connection heading the exact opposite direction, node B to node A (Figure 3). This structure could be used to form memory cells.

The overall structure of the network defines its functionality. From this, one can calculate the fitness score of the individual. Modifications of the networks are made through means of genetic mutations. The IO table, number of nodes, and node connectivity can be altered, generating new designs with new functionality. The genetic information is made up of numbers that represent devices made of nodes and connections. Negative numbers represent input and output nodes and positive numbers represent connections and transceivers.

Here is an example of DNA for two-half bit adder, which consists of four input nodes and two output nodes:

```
-2 -3 -4 -5 -6 -7 1000000000 1200111111
1000000000 1000000000 1000000000 -1 2 5 1 4
3 6 0 4 3 5 6 5 2 6 8 7 3 8 0 9 1 9 10 8 7 9
```

One cannot differentiate between input and output nodes in the DNA because they are predetermined. The numbers of input and output nodes depend on the type of adder being used². For example, a one-half bit adder consists of two input nodes and one output

²An adder has the same number of input nodes and output nodes. We use half-bit adder devices, which have half the number of outputs than the inputs. Half-bit adder devices shorten the time needed for evolution, and using multiple half-bit adders enable the occurrence of an epoch structure, and can have the general design illustrated below.

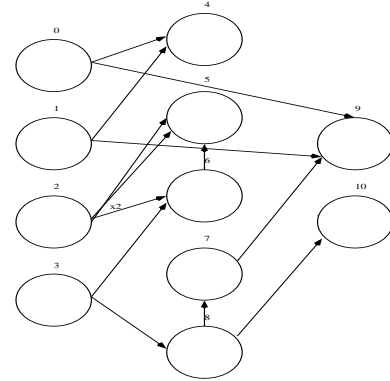


Figure 4: Sample network structure.

node whereas two-half bit adders have four input nodes and two output nodes. In the example above, the first four numbers (-2, -3, -4 and -5) represent input nodes and the following two numbers (-6,-7) represent output nodes. The five long strings of numbers that follow the input and output nodes are the transceivers that process the numerical signals. The -1 value in the DNA separates the node definitions and the connection list. Nodes in the network are numbered beginning with zero to the number of nodes n-1. The input nodes are numbered first, followed by the output nodes, and finally the firing nodes. The connection list consists of pairs of numbers that represent nodes; the connection goes from the node corresponding to the first number of the pair to the node corresponding to the second number. For example, 2 is connected to 5, 1 is connected to 4, and 3 is connected to 6. In Figure 4, we depict the network defined by the DNA above.

3.2 Data

We examine the existence of evolutionary acceleration by utilizing our evolutionary simulation. The toroidal lattice is a 100x100 grid of cells, each of which contains a single evolving element. We initialize the grid with a single individual consisting of only input and output nodes, bereft of connections. Individuals reproduce with a frequency which is determined by a fitness value which determines how well the individual solves the desired design problem; more fit individuals have the opportunity to reproduce more quickly. Individuals are subject to mutations only during reproductions. Each mutation event is limited to a single mutation during which a connection may be changed, a node's functionality altered, connections or nodes added or



deleted, etc. Multiple mutations require multiple mutation events. We evolve three different devices: single half adders, two half adders, and three half adders. The multiple half adders are simply multiple devices evolved simultaneously.

Each run is carried out on an individual PC class computer running Linux. The computer ranges in speed from 900 MHz to 1.3 GHz. All simulations are run to completion, which requires an increasing amount of computational time. For the devices reported here, the computation time ranges between less than one second for the one half bit to several hours for the three half bit devices. Increasing computation time prevents the extension of the computation to more sophisticated devices.

We calculate the average theoretical time needed to evolve a desired device by obtaining the sum of the inverse of the probabilities of reaching each new epoch. For each step in the history recorded for each epoch, a probability of reaching the next evolutionary advancement is calculated. At each mutation step, a network will randomly select a mutation out of a pool. Based upon the mutation’s properties and its chance of being selected, the probability of having chosen this mutation and its particular action out of all of the mutations and its possible outcomes are calculated.

As an example, consider the case of the simulation randomly selecting a mutation that adds a random connection to a selected network and connecting some node A to some node B. In order for this to occur, the mutation must be chosen out of all available mutations, and it must select two random nodes out of the network. The product of the probabilities of choosing the particular connection mutation and choosing the correct nodes is the overall probability of the mutation.

At the end of each improvement step, one or more of these actions will have been taken. The product of all of these probabilities forms the probability of the improvement having happened. The inverse of this probability is the expected time of the improvement event. This together with the actual evolution time, forms the data key part of evolutionary acceleration is that we add all these calculated combined probabilities at the end, as opposed to multiplying them. The inverse of this final number is considered the time of the evolution event.

The calculated sum of inverse probabilities and the actual evolution time in iterations are represented graphically in Figure 5.

Both the average and standard deviation of the different devices’ evolution times and summed inverse prob-

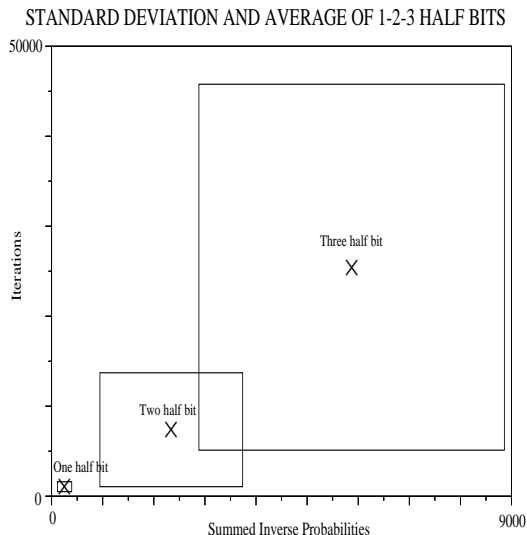


Figure 5: This figure illustrates the average and standard deviation of the evolution times and summed inverse probabilities of multiple half-adder-based devices.

abilities fall into different ranges in the graph. This underscores the tacit assumption that different devices, particularly multiple identical devices, will have differing average behaviors. The Figure illustrates a remarkable linearity in the data. Though the data does seem to be slightly superlinear, it is statistically consistent with a linear relation. What this would seem to indicate is that the increase in computation time does occur according to the expected dynamics. Were the product of the probabilities dominant, the graph would be significantly superlinear. The absence of superlinearity would seem to indicate that the NEA is a dominant effect in these simulations.

4 Concluding Remarks

This study has examined the occurrence of numerical evolutionary acceleration that naturally arises in evolutionary systems as a result of the continual series of epochs that occur in constrained evolving systems. We have examined the phenomenon in the context of a simple evolving system. In this system, we have explored evidence for the existence of the phenomenon, which yields significant hope for the development of a method for the design and implementation of evolutionary design systems.

The import of this method lies in its ability to make overall complex evolutionary processes possible despite a number of evolutionary steps that must be undertaken together. The existence of epochs allows popu-

lations to build off of the discoveries of any members. This makes the overall design process a set of punctuated steps in which mass extinctions happen one after the other. Yet it is only with populations of individuals that this effect can take place, for it is only large numbers of test cases which yield the expected dynamics. Any single individual evolving, or small population, suffers from the limitation of the number of test cases. On the other hand, the fact that the population is continually producing new individuals which must compete with one another makes the long-time population virtually unbounded, which allows small populations to be overcome.

What this study would seem to indicate is that as an *engineering specification*, the design of evolutionary systems which utilize a complete epoch series is necessary to have the additive quality for structural evolution. Mathematically, this has a significant advantage over systems which do not use it. In past studies, [Kazadi, 1997,98,99,2001,2003] we have concentrated on the development of compartmental models, which is itself an important part of the evolutionary system. However, this study would seem to indicate that an *equally important* part of the evolutionary design is the ability to use NEA.

In future studies, it may also be possible to examine the possibility of NEA having had an effect in natural evolution. As an example, it may well be the case that compartmentalization and NEA may actually be a factor in the explosion of species in the Pre-Cambrian Era. Since the compartmentalization of DNA systems allow individual traits to be evolved independently, and NEA would allow for the rapid development of each of these traits, it may be that the great push to speciation was facilitated by NEA and also by correctly compartmentalized DNA. This possibility warrants future study.

5 Acknowledgements

This work was completed at the Jisan Research Institute. One of the authors, S. Cheung, is supported by a generous grant from the California Institute of Technology Center for Neuromorphic Engineering, which in turn is supported by the National Science Foundation grant number EEC-9402726.

References

[1] S. Kazadi et. al. *Design of an Evolutionary Pre-processor*. Submitted, 2003.

- [2] S. Kazadi, H. Lin, P. Hung, D. Lee, D. Tsikata, J. Ogita, V. Huang. *Conjugate Schema in the HP Heteropolymer Model of Protein Folding and Protein Design*. **Complexity International**, 7, 1999.
- [3] S. Kazadi. *Conjugate Schema and Basis Representation of Crossover and Mutation*. **Evolutionary Computation**, 6(2): 129-160, 1998.
- [4] S. Kazadi, Y. Qi, I. Park, N. Huang, P. Hwu, B. Kwan, W. Lue, and H. Li. *Insufficiency of Piecewise Evolution*. **Proceedings of the Third NASA/DoD Workshop on Evolvable Hardware**, Long Beach, CA, 2001, pp. 223-231.
- [5] S. Kazadi, D. Lee, R. Modi, J. Sy, and W. Lue. *Levels of Compartmentalization in Artificial Life*, **Proceedings of Artificial Life VII**, M. Bedau, S. McCaskill, N. Packard, and S. Rasmussen, eds., Cambridge, Massachusetts: MIT Press, 81-89, 2000.
- [6] S. Kazadi. *Conjugate Schema in Genetic Search*. **Proceedings of the Seventh International Conference on Genetic Algorithms**, San Mateo, Ca: Morgan Kaufmann Publishers, pp. 10-17, 1997.