“TiPo” — A “Timed Pointer” Neural Net Model with Superior Evolvabilities for Implementation in a Second-Generation Brain-Building Machine BM2

Jonathan Dinerstein  
Sorenson Media, Inc.  
jon@sorenson.com  
1 (435) 792-1137

Hugo de Garis  
Utah State University  
degaris@cs.usu.edu  
1 (435) 512-1826

Brain Building Group, Utah State University, www.cs.usu.edu/~degaris

Abstract

For nearly a decade, the second author has been dreaming of building artificial brains. This involves the evolution of neural net circuits at electronic speeds in dedicated evolvable hardware, and assembling tens of thousands of such individually evolved circuits into humanly defined artificial brain architectures. However, this approach will only work if the individual neural net modules have high evolvabilities (i.e. the capacity to evolve desired functionalities, both qualitative and quantitative). This paper introduces TiPo (pronounced “Type-Oh”), a new neural net model with superior evolvabilities than all previously known neural net models. This new model is especially effective at curve following, which has practical applications in robot motion control. Our goal is to implement this model in a second-generation brain-building machine BM2.

1 Introduction

For nearly a decade, the second author has been dreaming of building artificial brains by evolving neural net circuits (of some 1000 artificial neurons each) in special hardware at hardware speeds. In the first attempt to implement these ideas (1996-2001), a cellular automata-based neural net model called “CoDi” [1] was used. It was simple enough to be implementable, despite the severe constraints imposed by the programmable (evolvable) state of the art hardware of 1996 (the Xilinx 6200 family of FPGAs). The first generation brain-building machine [6], called “CBM” (CAM-Brain Machine, CAM = cellular automata machine) was implemented and sold in Japan, Europe, and the US (4 machines). It is a research machine and needs further work. However, its chips date from 1996 and are starting to age. Independently of whether or not the CBM research and development work will continue or not, it is time to start thinking about the creation of a second-generation machine, which we call “BM2” (Brain-Building Machine, 2nd generation).

The basic idea behind brain building is to evolve neural net circuits at electronic speeds (about one evolved neural net in a few seconds). This elite evolved neural net can then be downloaded into a gigabyte of RAM. Tens of thousands of such individually evolved neural net “modules” can be downloaded. Human “brain architects” (BAs) then interconnect these modules to make human-specified artificial brain architectures. The same machine that was used to evolve the individual models (the CBM), is then employed to perform the signaling of the assembled RAM-based artificial brain in real-time. The brain can then be used, for example, to control the behavior of a robot and to give it thousands of different pattern detector circuits, decision-making circuits, and hundreds of motion control circuits. This is the dream. Modern electronics makes it possible, and the probability that Moore’s law will continue for another 20 years or more, makes the prospect of brain building almost certain.

However, if one is to perform “evolutionary engineering” (i.e. the construction of complex systems using an evolutionary (Darwinian) approach) then it is critical that the systems that evolve are capable of performing the tasks required of them. The researchers in the field of evolutionary engineering have become increasingly conscious over the past few years of the importance of “evolvability”, i.e. the ability of the systems being evolved to actually perform the task required of them. Since there is no theory of evolvability, an empirical “hit or miss” approach is used. As any evolutionary engineer (“EE”) knows, there is no certainty that a desired functionality will actually evolve. Even if the desired function does evolve, it may not evolve well (i.e. with high quality). The second author has coined the term MEC (Modular Evolvable Capacity) [6] to express a quantitative measure of the quality of the evolution, within a given
context. For example, for how long can an emitted analog signal from an evolved neural net follow some time varying target signal before it inevitably diverges. We have become increasingly aware that neural net models need to be chosen which prove to have high evolvabilities, so that they may perform well when implemented in programmable/evolvable electronics.

Of course, some neural net models will outperform other models for certain applications. For this reason, it is interesting to support more than one neural net model in the second-generation brain-building machine. Further, it is important to develop neural net models that can effectively perform critical functions, such as motion control and pattern recognition, for artificial brains.

This paper presents a new neural net model with evolvability levels superior to any previously known neural net model. This new model is especially effective at curve following, which has practical applications in robot motion control (e.g. sending time varying control signals to a robot’s motors, to generate different behaviors). Our goal is to implement this model (and perhaps other models) in a second-generation brain-building machine BM2.

The contents of the remainder of this paper are as follows. Section 2 briefly describes the neural net models (GenNet and DePo) that our new model (TiPo) is based upon. Section 3 introduces our new neural net model (TiPo). Section 4 presents experimental results concerning the TiPo model. Section 5 contains some discussion and plans for further research.

2 The GenNet and DePo Models

2.1 GenNet

The GenNet neural net model was introduced in 1990 [3]. It is a fully connected, feedback network evolved under genetic control. It is fundamentally quite similar to other traditional feedback neural net models. The GenNet chromosome is made up of \( N^2 \) weights, where \( N \) is the number of neurons. The weights and signals are signed real numbers, of the form \( +/−0.yyyy \) where yyyy is a binary fraction. The weight values lie in the range \( 0 \leq |W| < 1 \).

GenNet has two genetic operators:

1. Change one bit of a weight.
2. Change the sign of a weight.

GenNet uses a traditional activation function (equation 1), and sigmoid signaling function (equation 2). In other words, for each neuron, the dot product of the incoming signals and associated weights is computed. The output signal is then transformed using the sigmoid signaling function before being transmitted.

\[
S_j = \sum_{i=1}^{N} W_{ij} * S_i
\]  

(1)

\[
f(S_j) = \frac{2}{1 + e^{-S_j}} - 1
\]  

(2)

GenNet (as well as other traditional feedback neural net models) suffers from some significant weaknesses. Since the network is fully connected, every neuron receives a signal from and transmits a signal to every other neuron (including itself). When the number of incoming signals is large, the evolvability of the neural network may actually be worse than networks with few incoming signals, even though this means there are fewer neurons (and therefore less computational potential). We have observed this behavior with applications that have dynamic input and/or output signals, e.g. curve following and dynamic pattern detection.

Even with a reasonable number of neurons (e.g. \( N = 20 \)), GenNet is still a poor neural net model for curve following. This is shown in figure 1. Curve following has importance in brain building for motion control, and the evolvability of GenNet for this function is not sufficiently accurate to be useful. We will later show the superior evolvability and hence curve following quality achieved by DePo and TiPo networks when following the same target curve, whose formula is:

\[
y(x) = 0.52 + 0.25\sin(2\pi x / 60) - 0.15\cos(2\pi x / 40) + 0.1\sin(2\pi x / 50).
\]  

(3)

![Figure 1. Output of a fully evolved GenNet neural network, trained to follow a curve 150 clock ticks in length. As can be seen, GenNet does not follow the target curve particularly well.](image)
Despite weak curve following, GenNet has proven effective at pattern recognition (a traditional strength of feedback neural networks). As a result of our experimentation, we believe that GenNet is generally more effective at processing static input and output signals than dynamic signals. Thus, we have focused the design of DePo and TiPo towards increased evolvability for functions involving dynamic signals.

2.2 DePo

We recently introduced the DePo (Delayed Pointer) neural net model [4,5]. The design goal of DePo was to increase evolvability levels relative to traditional feedback neural net models (such as GenNet), especially for functions involving dynamic signals (input and/or output). To accomplish our goals, we believed it necessary to have time-sensitive functionality in our new neural net model. Consider curve following: a curve is fundamentally a function of time (or some other similar parameter). We theorized that a superior curve-following neural net model would have some “understanding” of time (i.e. a functionality dependent on time with associated genetic operators). Another major design goal of DePo was to avoid the evolution barrier for nets with many neurons (signal overload, i.e. if too many signals come in, the activation value becomes so large that the sigmoid signal value is pushed very close to +/-1, giving the evolutionary algorithm very little room for manoeuvre).

As the name “DePo” suggests, its two essential features are its use of pointers (to the other neurons that an individual neuron connects to), and delays (the number of clock ticks it takes for a neural signal to be transmitted from the emitting neuron to the receiving neuron). Both pointers and delays are under genetic control. Weights and delays in DePo are of the same format as GenNet (see section 2.1). Also, the same traditional activation and sigmoid signaling functions are used (equations 1 and 2).

The concept of the pointers is fairly straightforward. They reduce the neural net from being fully connected (such as GenNet) to partially connected. This reduces the number of signals being received by each neuron, making it possible to evolve higher-quality neural nets with many neurons.

The delays can be thought of as queues. In other words, a link between two neurons is a queue of some independent length, and signals pass through the queue at a rate of one cell per clock tick. The length of a queue is the delay of that link. When initializing the neural network, the signal queues are clear to zero. Thus, no non-zero signal arrives to the destination neuron until the first emitted signal passes through the entire queue.

The DePo chromosome is composed of two subfields, the weights of the links and the delays of the links. If the network is fully connected, there are $N^2$ weights and $N^2$ delays. The genetic operators of DePo are:

1. Change one bit of a weight.
2. Change the sign of a weight.
3. Increment a delay.
4. Decrement a delay.
5. Create a new random delay.
6. Add a neuron (random links to other neurons are created).
7. Delete a neuron (all associated links are also created).
8. Add a link (a random weight and delay are assigned).
9. Delete a link.

DePo has proven highly evolvable for many applications, especially those involving dynamic signals, as can be seen in figure 2. DePo proved to be greatly superior to GenNet in curve following (c.f. figure 1).

![Figure 2. Output of a fully evolved DePo neural network, trained to follow a curve 150 clock ticks in length. As can be seen, DePo’s performance for curve following is greatly superior to GenNet’s. However, the curve following is still not perfect.](image)

We also found DePo to be very effective in dynamic pattern recognition. To test this, we performed an experiment where a DePo neural net was evolved to recognize a series of input movies, based on the speed of motion in the movies. Admittedly the evolution required several hundred thousand generations, but once the network evolved, it performed very well.

The most interesting aspect of DePo is the delays. The delays make it possible for signals to travel through time unaltered. In a sense, this provides a form of long short-term memory [2]. Also, according to computer theory, memory is a requirement for increased
computational capacity. Our research thus far has only scratched the surface of potential applications for DePo, and we are confident that it will perform well for other functions.

While DePo has performed well for curve following, it can be seen in figure 2 that its ability for curve following is not perfect. Therefore, for curve following, we have developed a further neural net model called “TiPo” (pronounced “Type-Oh”).

3 The TiPo (Timed Pointer) Neural Net Model

TiPo is a new neural net model with a truly amazing evolvable capacity for curve following. Before designing TiPo, we analyzed what made DePo better at curve following than GenNet. We discovered that DePo’s delays provided the greatest benefit, but not in the way we expected. As discussed in section 2.2, the signal queues are initially cleared to zero, and transmitted signals do not arrive until they have passed through the entire queue. Therefore, no “interesting” signals arrive over a link until the delay period has been passed. This naturally reduces the number of signals being received at each neuron, making greater evolvability possible.

Based on this discovery, we designed TiPo to further evolve the time period in which signals are transmitted on each link. We did this by specifying a start and end transmission time for each link, which are under genetic control. The start time specifies the clock tick when the link will begin transmitting signals, and the end time specifies the clock tick when the link will stop transmitting signals. The signals are not delayed, i.e. they arrive immediately on the next iteration of the neural net.

The TiPo chromosome is composed of three subfields: weights, start times, and end times. There is one weight, start time, and end time associated with each link. The start and end times for a link are not allowed to cross (i.e. start ≤ end), nor pass beyond the time range of the target signal being trained for (0 → T clock ticks). The genetic operators of TiPo are:

1. Change one bit of a weight.
2. Change the sign of a weight.
3. Increment a start time.
4. Decrement a start time.
5. Create a new random start time.
6. Increment an end time.
7. Decrement an end time.
8. Create a new random end time.
9. Add a neuron (random links to other neurons are created).
10. Delete a neuron (all associated links are also deleted).
11. Add a link (a random weight, start time, and end time are assigned).
12. Delete a link.

The results of TiPo for curve following are truly impressive, as shown in figures 3 through 5, as well as in figure 8 (at the end of the paper) which depicts in fine detail the quality of the evolved TiPo output curve.

We also achieved promising results evolving TiPo to follow a curve containing sharp “points” (figure 6). In other words, the function defining the target curve has sudden large changes in its first derivative. Feedback neural nets generally output smooth curves (in both the first and second derivatives), and therefore have difficulty following the sudden changes in this target. As can be seen in figure 6, GenNet does not reach any of the points on the top or bottom of the target curve. However, TiPo matches the target curve much more closely, as shown in figure 7.

Figure 3. Output of a fully evolved TiPo neural network, trained to follow a curve 150 clock ticks in length. As can be seen, TiPo’s performance for curve following is virtually flawless. Also of importance, was the fact that this impressive curve-following was accomplished with very few neurons (N = 20). This network was evolved in about 30 minutes.

4 Experimental Results with TiPo

As shown in figures 3 through 6, TiPo is amazingly effective at curve following, even when using a small number of neurons. This potential to operate well with few neurons has importance for hardware implementations (such as in a brain-building machine), as every extra neuron and link used requires extra number of gates in an FPGA.
Figure 4. Output of a fully evolved TiPo neural network, trained to follow a curve 400 clock ticks in length. Note that the target curve was generated using the same function as in figures 1-3 (equation 3). As with the 150 clock tick experiment shown in figure 3, this longer output curve is virtually flawless. Note that more neurons were used (N = 40), but this is still not a large number.

Figure 5. Output of a TiPo neural net for 1000 clock ticks. The target curve is defined by equation 3. Even for such a long target curve, the TiPo output curve is still virtually flawless. This network was evolved in a few hundred thousand generations, taking about eight hours on a 2.2 GHz Pentium 4 computer.

It is interesting to note that for difficult problems (for which there can never be a perfect fitness, such as curve following), TiPo will add more and more neurons until the user-prespecified maximum number is reached. The neurons are added rapidly at first (when there are few neurons), but are added much more slowly later on (when there are many neurons). The fitness of the evolved neural net does improve with more neurons, but this improvement follows the law of diminishing returns (i.e. the more neurons, the less the benefit of adding more neurons). Furthermore, using fewer neurons does not significantly impact the quality of the neural net, unless a very small number is used. For example, every experiment we tried could be performed well with 20 neurons, although a greater number of neurons did improve results somewhat.

Figure 6. GenNet, evolved to follow a target curve with sharp “points.” As can be seen, GenNet does not reach any of the points on the top or bottom.

Figure 7. TiPo, evolved to follow a target curve with sharp “points.” Compare with figure 6. As can be seen, TiPo’s performance is greatly superior. However, it still doesn’t exactly match the sharp points of the target.

Another interesting point is that TiPo used most of the available links in our experiments (the evolved neural nets used about 80% of available links on average). In contrast, DePo used about 50% on average. We believe that TiPo uses more links because the start/stop timing naturally limits how many signals will arrive at a given neuron on each clock tick. Therefore, more links can be used without getting “signal overload.”

One advantage that TiPo has over DePo is that delays are not used. This means that there are no signal queues, and therefore lesser electronic storage requirements. Therefore, a hardware implementation of TiPo would be cheaper/simpler than a hardware implementation of DePo. However, this extra cost of hardware implementation could be worthwhile if it is shown that DePo outperforms TiPo (and any other
neural net model) for a key functionality required in brain building.

We also performed experiments to discover the minimum number of bits required for weights and signals before evolvability would break down considerably. This is an important topic for the field of evolvable hardware, because the more bits that are used, the more gates a (programmable) hardware implementation requires. For example, CoDi an earliest cellular automata based neural net model [1], was limited to 1-bit weights, 1-bit signals (with 1-bit signs +/-). We found that TiPo scaled very nicely for different numbers of bits. For example, we ran an experiment where weights and weights were limited to 4 bits plus a sign bit. The result is shown in figure 11. Even limited in this way, the quality of the neural net’s output is quite good. We tried using less than 4 bits, but evolution failed in our initial experiments. It seems 4 bits is the lower limit, a figure which should be readily implementable in state of the art programmable/evolvable hardware (e.g. Xilinx’s Virtex family of FPGAs, which contain up to 8 Megagates in a single chip).

We have yet to undertake further experimentation using TiPo for other important applications of neural nets in brain building, such as pattern recognition (static and dynamic), and decision-making. While we are not sure how well it will perform for static pattern recognition, we expect that it will be excellent for dynamic pattern recognition (much like DePo). It will be interesting to see if TiPo is better in general than DePo for brain-building functions, as its hardware implementation is simpler (as discussed previously).

One final topic we feel is important to discuss is the fact that GenNet (or another traditional feedback neural net) is simply a special case of a TiPo or DePo neural net. For example, if a TiPo neural net has all links active for the entire execution of the neural net, it is just a GenNet. Also, if a DePo neural net has all delay lengths set to zero, it is simply a GenNet. As for the pointers, this same architecture (not fully connected) can be achieved with GenNet by evolving weights to be zero. The fact that TiPo and DePo can both be evolved to be GenNets, suggests to us that these new models may be able to perform the same tasks as GenNet to at least an equal quality. As traditional feedback neural net models (such as GenNet) are known for being good static pattern recognizers, we expect this will also be true for TiPo and DePo. However, we have not yet proven that a TiPo or DePo network will actually evolve to be a GenNet network (if it is the optimal architecture for the function being evolved).

5 Discussion and Future Research

Despite our great pleasure at witnessing the considerably superior evolvability of the new TiPo model, a lot of work remains to be done on it to discover its strengths and limitations. Once this has been done, the next stage in our research is to determine what neural net model(s) will be most effective in brain building. This may involve developing further neural net models. Finally, once we have selected the neural net model(s) to use, we will begin the translation of the (modified) model into programmable/evolvable hardware to start building the second-generation brain-building machine BM2 (funding permitting). It will be interesting to see to what extent we will be forced to compromise the TiPo model to fit it into state of the art programmable chips, although the 4-bit weight and signal lower limit discussed in the previous section suggests that such a compromise will not be severe. If future research shows that pattern detection and decision making circuits (e.g. circuits evolved to implement production rules) can be evolved with similar quality levels as shown by the TiPo model, these models will be implemented in the second generation brain building machine BM2. Once it can be shown that single neural circuit modules can be successfully evolved and implemented in evolvable hardware, multimodule experiments can be undertaken in minutes in hardware, rather than days or weeks in software simulation, to gain experience in architecting artificial brains.

It remains a decade long dream of the second author to see not only the creation of a new research field called “brain building”, but to see that new field expand sufficiently to create a brain building industry. The fact that programmable chips have already nearly 10 million gates, and that Moore’s Law is likely to hold for another 20 years, makes the achievement of such a dream a virtual certainty. The first artificial brains will very probably be built within the next few years. If the dotcom crashed had not bankrupted the second author’s research lab and killed the first generation brain building machine (CBM) attempt, the first such brain might have been a reality already.

References


![Figure 8]( Attached Image )

**Figure 8.** This is the same graph as figure 5, just at a larger scale to show fine detail. TiPo follows the target curve almost exactly, except for the extreme peaks of the target signal. This is because output values near 1 are very hard to reach, due to the sigmoid activation function we are using.
Figure 9. This is the same curve as in figure 8, but the output signal is offset by 0.1. This offset is necessary to see the fine detail of the output signal, as it is an excellent approximation of the target curve.

Figure 10. This graph shows how TiPo will gracefully break down when too much is required of too few neurons. The same number of neurons (N = 50) are used for this 3000 tick curve as for the 1000 tick curve in figure 8. Note that those regions of the target curve that are very difficult to follow (such as the minima and maxima) are not followed.
Figure 11. This graph shows TiPo with 4-bit signed weights and signals. As can be seen, the output signal is quite discrete. However, the TiPo net does an excellent job at following the target curve. Based on these 4-bit results, an evolvable hardware implementation of TiPo appears very promising. (All other experiments in this paper used 12-bit weights and 32-bit signals).