Genetic Generation of Connection Patterns for a Dynamic Artificial Neural Network

John G. Elias

Department of Electrical Engineering
University of Delaware
Newark, DE. 19716

Abstract

Work-in-progress on the use of a specialized genetic algorithm for training a new type of dynamic artificial neural network is described. The network architecture is completely specified by a list of addresses that are used to connect signal sources to specific artificial synapses, which have both a temporal and spatial significance. The number of different connection patterns is a combinational problem which grows factorially as the number of artificial synapses in the network and the number of sensor elements increases. The network is implemented primarily in analog electronic hardware and constructed from artificial dendritic trees which exhibit a spatiotemporal processing capability that is modeled after morphologically complex biological neurons. We describe work-in-progress on using the specialized genetic algorithm, which has an embedded optimizer in place of the standard mutation operator, for training a dynamic neural network to follow the position of a target moving across an image sensor array.

1 Introduction

The artificial neural network discussed in this paper is unlike most other neural network architectures that have been described before. It is not a multilayer perceptron (Rosenblatt 1962), it is not a Hopfield network (Hopfield 1982), nor is it like any other architecture known to the author. However, since practically all types of artificial neural network architectures were modeled, with varying levels of abstraction, after biological neurons, there are common ideas that are used in all. The main difference comes from the approach taken in modeling important biological structures. In previous artificial neuron models, the neuron has been treated as a point entity that receives and processes inputs at the soma (cell body). This approach resulted in the perceptron, which is a linear combiner with a nonlinear thresholding output unit. In our modeling approach, we have looked beyond the soma to the extensive dendritic tree structure of neurons, which not only forms most of the cell's surface area but provides the spatiotemporal signal processing capabilities not present in models which assumed a point-entity neuron.

Artificial dendritic trees (Elias 1992) are analog circuits that are highly sensitive to both temporal and spatial signal characteristics. Artificial dendritic trees do not make use of the conventional neural network concept of weights, and as such do not use multipliers, adders, look-up-tables, or other complex computational units to process signals. They do, however, support a large and arbitrarily high-precision signal-sensitivity space which is due to an extremely large number of spatial connection patterns and to the sublinear electrical behavior of the artificial dendritic tree. Therefore, the weights of conventional neural networks, which take the form of numerical, resistive, voltage, or current values, but do not have any spatial or temporal content, are replaced with connections whose spatial location has both a temporal and a weighting significance.

We have just begun researching this new type of artificial neural network built with dendritic trees. We have implemented our architecture in CMOS VLSI (Elias et al. 1992), and we have trained a simple network to recognize static images using a genetic algorithm similar to the one described in this paper (Elias and Chang 1992). However, our primary interest is in time dependent signal classification and system control, which places greater demands on the training method and on the dynamic properties of the network. We have chosen a simple time dependent target tracking application for evaluating the efficacy of various forms of genetic algorithms. Before discussing the test application and the preliminary results of using a genetic algorithm for training, we provide some background information on our dynamic neural network and on the biological dendritic trees from which it is modeled.

2 Spatiotemporal Properties of Dendritic Trees

Figure 1a is a drawing of a typical Purkinje neuron from the cerebellum and attempts to show the extensive dendritic tree structure that is typical of most neurons. Neuron anatomy can be generalized as having three major parts with the following highly simplified functional descriptions: 1) a spatially extensive dendritic tree, which collects and processes the majority of afferent signals; 2) a soma, which forms a response based on the collective dendritic tree electrical signal; and 3) an axon, which propagates efferent impulsive signals from the soma to distant dendritic trees of other neurons. A large fraction of the surface area of the Purkinje cell forms the dendritic tree, which in humans supports ~10^5 synaptic sites.

Figure 1b is a drawing of a small section of dendrite which can be modeled as an electrical circuit with a membrane capacitance, C_m, in parallel with a membrane resistance, R_m, and a series axial cytoplasmic resistance, R_a. A linear second-order differential equation, shown in figure 1c, describes the one-dimensional voltage profile for a given current density, I(x,t). If constant current is injected at a particular point along the dendrite, the voltage profile decays exponentially with respect to the axial distance from the site of injection. Therefore, the voltage, when measured at a particular point, shows a nonlinear relationship to the spatial location of current injection along the dendrite. This fact provides a means to scale or weight an input current signal, over a wide dynamic range, by simply changing the spatial position of its connection to different synaptic sites along the dendrite.

An important property of passive dendritic trees is illustrated in figure 1b, which shows four afferent signals forming synapses at sites *A*, *B*, *C*, and *D*, which, in this drawing, are on knob-like

processes called spines. For synaptic events that are temporally coincident and that are located at electrically nearby sites, the resultant signal measured at the soma is a sublinear function. Conversely, for temporally coincident signals in which the synaptic sites are electrically distant from each other, the resultant signal is nearly linear. For example, a sublinear resultant voltage is measured at the soma if sites *A* and *B* are simultaneously active in figure 1b. However, if sites *A* and *D* are simultaneously stimulated the resultant voltage at the soma is nearly linear. This phenomenon is extremely important in providing a rich environment for computation.

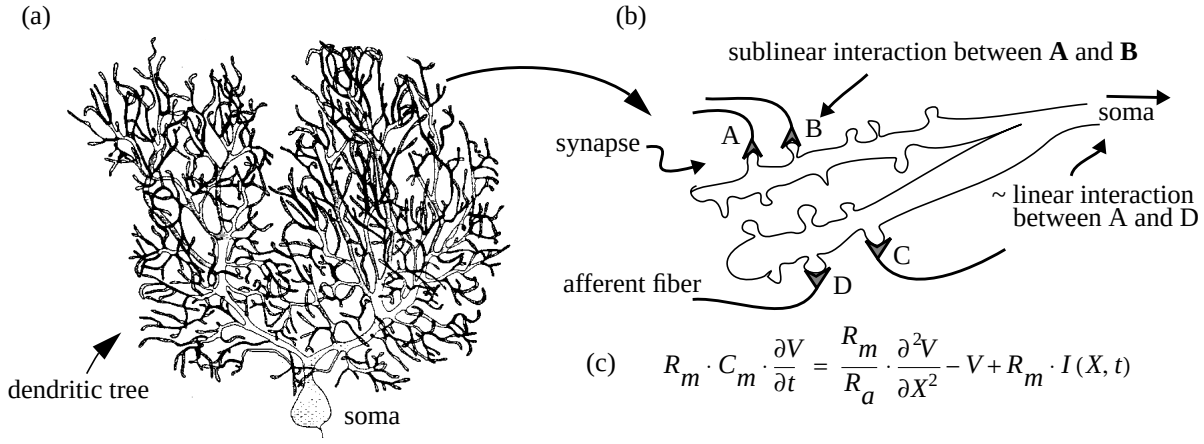

Figure 1: a) Drawing of typical Purkinje cell, after Berry and Bradley (1976). b) A drawing of two branches of a dendritic tree showing the microcircuit structure which includes knob-like projections called spines. Synaptic sites near each other exhibit sublinear interactions (e.g. sites A and B). Synaptic sites that are widely separated, however, interact much more linearly (e.g. sites A and D). c) One-dimensional cable equation which gives the membrane voltage, V, at location X at time t as a result of current injection, I(X,t), C_m is the membrane capacitance in parallel with a membrane resistance, R_m. R_a is a series axial cytoplasmic resistance.

The transient response of dendrites is of particular value since we are interested in processing dynamic signals. Figure 2a depicts a simple dendritic tree to illustrate the transient response due to impulse current injected at four different locations, *A*, *B*, *C*, and *D*. The voltage impulse response as measured at the soma, which, for this case is a simple lumped circuit with no active elements, is shown in figure 2b. Only one site on the dendrite is stimulated at a time. Therefore, figure 2b was created by overlapping four different impulse responses. Three important features of the impulse responses shown in the figure are worth discussing. First, the peak voltage amplitude is larger for stimulus sites nearer the soma and gets progressively smaller for sites further away. Second, the time at which the peak occurs shows a similar dependency on the distance from the injection site so that distal sites are spread out in time and reach their peak values later than more proximal sites. Third, the asymptotic transient behavior is independent of site location provided that the membrane capacitance and resistance are constant. This transient behavior suggests the possibility of powerful dynamic signal processing capabilities using simple circuit elements that are modeled after dendritic trees. Networks that are built using these tree structures can process spatiotemporal signals by mapping afferent signals to specific locations on the artificial dendritic tree.

The electrical response due to synaptic stimulation either diffuses or propagates to all other portions of the cell interior. As the electrical signal spreads to other parts of the cell, an action potential or spike might be generated (e.g. at the soma). An action potential is a highly nonlinear voltage transition that is usually triggered by the membrane voltage exceeding some threshold value. It is most often associated with axon signal propagation to distant neurons. An excitatory response is classified as one that tends to increase the likelihood of generating an action potential, either somatic or dendritic, and it usually has a positive voltage trajectory. An inhibitory response often has a negative voltage trajectory and tends to decrease the probability of subsequent action potential generation.

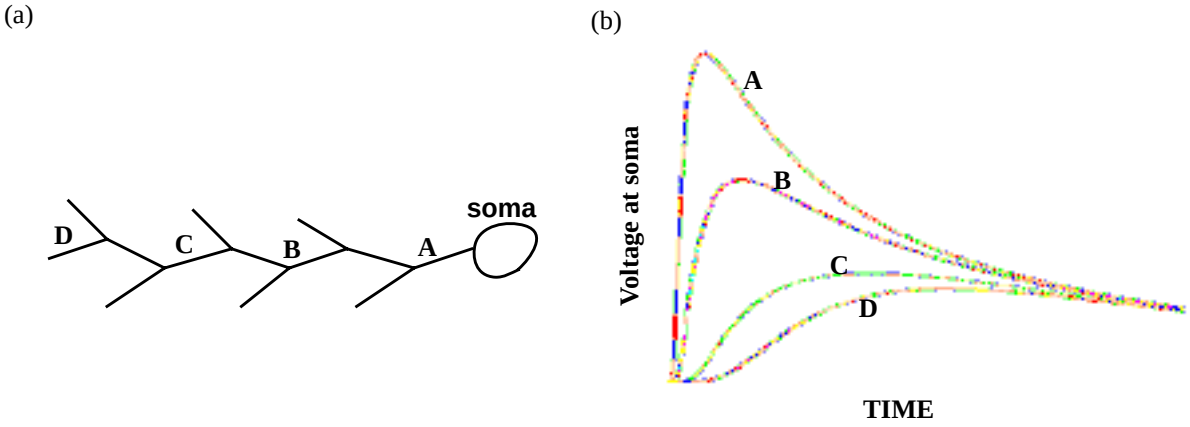

Figure 2: a) Simple branching structure to illustrate transient response of passive dendritic trees to impulse current injected at various sites along a branch. The resultant voltage is recorded at the soma, which, in this case, is a passive element. b) Overlapped transient voltage responses for impulse current injected at sites A, B, C, and D. Both the time at which a peak occurs and the peak amplitude is a nonlinear function of the position of the injection site with respect to the soma.

3 Networks with Artificial Dendritic Trees

Our networks comprised of artificial dendritic trees operate as illustrated in figure 3. In this example, a 2-dimensional sensor array (e.g. a CCD) supplies information to an artificial dendritic tree through two sets of digital memories and a state machine which act, in effect, like connecting wires. These virtual wires provide a simple way to change connection patterns during training. The first memory set is a multiple-bit-wide doubly-linked list which holds the artificial synapse addresses that each sensor element connects to. We call this memory the Connection List. It is the Connection List which is changed during training. The second memory is a single-bit-wide register which physically connects to a synaptic site which effects either an excitatory response or an inhibitory response. Each artificial synapse has one register associated with it. We call the collection of registers the Stimulus Memory. The contents of the Stimulus Memory is a function of the Connection List and the current sensor element states. During network operation, if the Stimulus Memory for a particular artificial synapse has been activated then that synapse is turned on when the STIMULATE signal is asserted (Elias et al. 1992).

The sensor is a thresholding device which outputs ones or zeros. It periodically collects visual data from some image field of interest and through the virtual wires supplies this information in parallel and impulsively to the synapses of the artificial neuron. Each artificial neuron in the network responds to the afferent sensory signals by producing an impulsive output which either becomes an input to other neurons in the system or it is used in the control of an effector, such as a motor that moves the sensor to point at a new image field.

The artificial neurons in the system make connections through virtual wires with artificial synapses on other neurons, and just like sensor elements, their outputs are in one of two states: active or inactive. The outputs of the artificial neurons are sampled in the same way that the sensor elements are sampled, and the corresponding artificial synapses are activated if the neuron they are connected to is active. The active output state of an artificial neuron is changed to inactive after the state information has been collected by the state machine.

The system continuously cycles through the sensor elements and artificial neurons. A cycle begins by inactivating the Stimulus Memory (i. e. all locations are cleared). The sensor elements are read one at a time and, if active (i.e. shaded in figure 3), the artificial synapses listed in the Connection List for that sensor element are activated. The process is repeated for all sensor elements and for all artificial neurons in the system. After all sensor elements and artificial neuron outputs have been sampled, the activated artificial synapses in the system are turned on by asserting the STIMULATE signal.

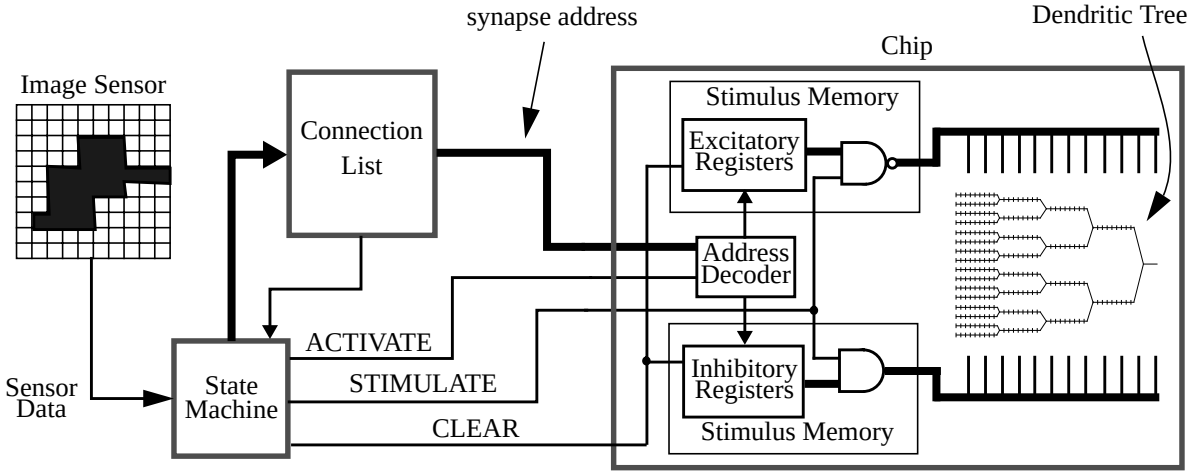

Figure 3: System block diagram which includes CMOS artificial dendritic tree chip. The state machine reads each sensor element one at a time. If a sensor element is active, the state machine activates all of the artificial synapses that connect to that sensor element by accessing the Connection List, which contains the synaptic addresses. The Stimulus Memory is cleared (by asserting CLEAR) before each cycle of reading the entire sensor array. The STIMULATE signal is asserted after the sensor has been completely accessed and all appropriate synapses have been activated. When the STIMULATE signal is asserted all activated excitatory and inhibitory synapses enable inward or outward impulsive current, respectively, at specific sites on the dendritic tree.

3.1 Training Artificial Dendritic Tree Networks

The number of different connection patterns between sensor elements and artificial synapses is quite large even for small networks and is a factorial function of the number of synapses and sensor elements. If we limit, for the moment, the number of connections that each sensor element can make to one, then the total number of different connection patterns is given by

$$\frac{N!}{(N-M)!} \tag{1}$$

where N is the number of artificial synapses and M is the number of sensor elements.

We have tried to closely model our artificial dendrites and synapses after their biological namesakes. Biological neuronal systems display divergence of efferent signals from both sensors and neurons: a given sensor or neuron will usually make multiple synapses with other neurons. Biological systems also exhibit the property of synaptic convergence: within a small area, there exist synapses from many different neurons or sensors. Both synaptic convergence and neuron/ sensor divergence are important factors that lead to a richer computational space. Therefore, we allow each sensor element or artificial neuron to connect to many different synapses (an example of divergence) and we allow each synapse to receive signals from many different sensor elements or other artificial neurons (an example of convergence). This tends to make the number of possible connection patterns much larger than that indicated by equation (1).

In general, practical imaging systems will have a large number of sensory inputs (10^3 - 10^6) and an even larger number of artificial synapses (10^4 - 10^9). Each sensor element must be able to connect to any one of the artificial synapses in the system. Because the network is a dynamic analog circuit, the flow of sensory information must be in parallel (i.e. sensory stimuli arrive at all synapses at the same time, unless specifically delayed). Virtual wires provide for parallel stimulation, straightforward reconfiguration, convergent and divergent connections, simple and efficient hardware implementation, and compatibility with genetic algorithms' crossover and mutation operators.

We use a standard supervised method to train our dynamic neural networks: sensor data from a training set is applied to the network, the output is compared to the desired response, and the network parameters are changed to reduce the difference between the output and the desired response. In perceptron type neural networks, the network parameter that is changed is the weight matrix, which might be in the form of a set of numbers, resistances, voltages, light intensities, or currents. There have been several studies done recently using genetic algorithms to find near-optimal weight matrices (Whitley et al. 1990) and to determine efficient network architectures (Koza and Rice 1991). For our dynamic networks, the network parameter that is changed during training is the Connection List, which is a list of synapse addresses that specifies the connection pattern for the entire network. The effect of this change is to move sensor and neuron output connections to physically different synaptic sites.

How to change the network parameters in order to minimize the difference between desired and actual network response depends on the particular optimization method that is used. Gradient optimization methods like backpropagation are not highly suitable for training artificial neural networks of the type described in this paper because of the difficulty of calculating derivatives. Other, nongradient optimization methods, such as simplex, hill-climbing, and Powell's (Brent

1973) are by themselves either not very effective, because they are inherently greedy, or they require an excessive amount of computation to ensure a near-optimum solution. Of all the nongradient search methods, genetic algorithms seem to be the most promising in terms of training effectiveness and compatibility with hardware. Therefore, we are investigating several different versions of genetic algorithms for training our dynamic neural networks.

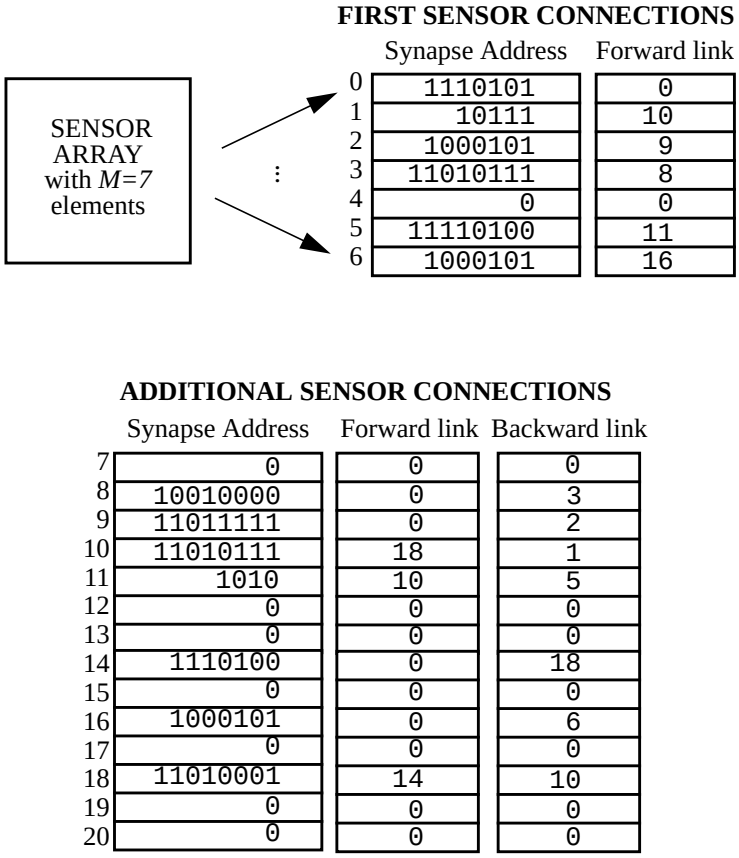

Figure 4: Connection List structure. The first *M* locations in list are pointed to by the sensor element addresses. These hold the first connection for each sensor element. The next K locations contain the addresses of additional connections for all of the sensor elements. The forward and backward links permit connections in the middle of list to be added or deleted. An entry of 0 for the forward link means no more connections for that sensor element. Only the synapse addresses undergo recombination during training.

The choice of string encoding representation is critical from both a system operation and a genetic algorithm point of view. The encoding scheme must produce an efficient, scalable, and suitable string for the crossover operation as well as be compatible with hardware system constraints. In our system, we specify the connections in terms of synaptic addresses for each sensor element and this information is stored in the Connection List. An address of zero represents a nonexistent connection for that sensor element. The first *M* locations of the

Connection List are pointed to by the sensor element address. In the example shown in figure 4, there are seven sensor elements, all but one (element 4) of which connect to physically distinct synaptic sites. An example of synaptic convergence is exhibited by sensor elements 2 and 6, which connect to the same synapse at location 1000101. The next *K* locations in the Connection List hold the addresses and links for additional connections. These memory locations are used to store divergent connections from sensor elements. However, not all sensor elements exhibit divergence. In figure 4, sensor element 1 has divergent connections to four artificial synapses, while sensor element 0 has no additional connections.

Thus far, we have only described the connection specification for sensor elements, but the outputs of artificial neurons are treated just like sensor elements in our system. Therefore, there is an identically structured Connection List that describes connections between artificial neuron outputs and the artificial synapses of other neurons in the system. It is uncertain at this time whether we will retain separate Connection Lists for sensor and artificial neuron outputs in the future. That decision will depend on results from our current research effort. The test application described in this paper makes use of just one artificial neuron. Therefore, the Connection List will only contain sensor element connections to the artificial synapses on that one neuron.

In our system, instances of the Connection List, each of which may hold a different set of connections, are the members of the population. The Connection List undergoes crossover and mutation operations in order to produce better performing offspring for the next generation. This is very much like what has been done with the weight matrix of perceptron-type networks in other studies (Whitley et al. 1990). Crossover is commonly done on encoded bit strings which results in a certain number of bits in both parent strings appearing in the offspring. We have decided to treat entries in the Connection List the same way bits are treated in standard genetic algorithms and use the crossover operator on connections rather than on the bits in the Connection List. With this approach, effective connections from each parent string eventually end up in the offspring. A similar strategy was developed by Montana and Davis (1989) for recombining real-value weight matrix elements.

While this particular implementation of the crossover operator is effective in terms of the hardware implementation, it presents somewhat of a problem because new artificial synapse addresses are not generated during recombination. The crossover operator recombines the existing connections but does not create new ones. Therefore, if connections that could improve the string's performance are not represented in the population of Connection Lists then we must rely on the mutation operator to produce them. However, mutation rates are generally kept low in order to preserve the useful strings in the population or when population diversity is already high enough. A low mutation rate means that new connection addresses might be produced too slowly for practical purposes. To solve this problem, we replaced the "standard" mutation operator with an optimization function that increases population diversity by evaluating new connections for each string and keeping only those that improve the string's fitness value.

We are evaluating several different methods for implementing the operations of selection and crossover. In previous work on a character recognition application (Elias and Chang 1992), the probability of selecting a string was based on the string's fitness value relative to all other strings, and only one crossover point was used. This approach worked well for that application, but for reasons not yet understood it did not work well for the dynamic problem discussed later in this paper. We had much better success solving the dynamic problem using methods borrowed from

Ackley (1987) and Whitley et al. (1990) for selection, crossover, and string retention.

The basic steps in our genetic algorithm are evaluation, selection, crossover, replacement, and optimization. During the evaluation step, the output voltage from the artificial dendritic tree is monitored for each Connection List string in the population. The peak voltage from each member of the population is used directly as the fitness value or in a function to produce the fitness value. The method used for calculating a fitness value has a direct effect on the convergence behavior of the genetic algorithm. For dynamic problems, we are evaluating the effects of including past and future expected values in the fitness function in order to speed up convergence.

Selection of mates is either done randomly or in proportion to a string's fitness value. Random selection follows Ackley's approach (1987) and is simpler to implement than fitness proportional selection (Holland 1975). For the work described here, the basic algorithm is to select two mates with uniform probability, combine them to produce two offspring, evaluate their fitness values, and replace a randomly selected member of the population which has below average fitness value with the better performing offspring, provided that the offspring's fitness value is above a certain acceptance threshold. Preliminary results indicate that this method is as effective as one that uses proportional selection, and it requires fewer computational steps. However, we will continue to evaluate the relative performance of random and fitness-proportional selection for larger dynamic networks.

The crossover operator, as we have implemented it, randomly picks two points and swaps string segments of the mates between these two points. Evaluation, selection, crossover, and replacement are done until population diversity becomes low, which can be indicated by the variance in the average fitness value for the population. When the crossover operation becomes ineffective because population diversity is too low, then a mutation operator is used to introduce new schemata. We have had good results using an optimizer in place of the conventional mutation operator. For the results reported here, the optimizer is based on a simulated annealing search algorithm (Kirkpatrick et al. 1983).

3.2 Results

A single branch of an artificial dendritic tree was used to test the efficacy of our version of a genetic algorithm. The sensory signals for each experiment come from a one dimensional thresholding sensor array, which might be a CCD imaging device or something like it. The sensor elements contain a logic zero if the image sensor field is above a fixed threshold voltage and a logic one otherwise. In each experiment, a sequence of sensor data over time is presented to the branch and the resultant waveforms are measured and evaluated to produce a fitness value. Convergence behavior depends on the fitness value evaluation. We found that satisfactory convergence was obtained when the fitness function was an exponential function of the difference between the desired and actual dendritic tree output.

Figure 5 shows an example of a single branch of artificial dendritic tree which can be used to provide a control signal for a target following application. In this example, a seventeen element sensor is used as the input device. Larger sensor arrays can be used to obtain similar results. The sensor pattern over time is that of a moving target, which, in this case, is a logic zero on a background of logic ones. However, the target could be of any shape as long as it was

distinguishable from the background, and the sensor could also be extended to two dimensions. When the target is on center, the output of the tree is zero. Small variations in target position around the center produce relatively small output voltage transitions useful for low gain system control. If the target moves below center, as shown in figure 5, the resultant voltage transients are positive. If, however, the target moves above center the transients are negative. As the target moves farther off center, either up or down, the resultant branch output voltage gets progressively larger. This occurs because more proximal artificial synapses become active, which, in effect, shifts the system control to higher gain. The relative amplitudes of the branch output voltage transients as a function of the distance between target and sensor center can be arbitrarily set by moving connections of particular sensor elements to either more distal or more proximal artificial synapses.

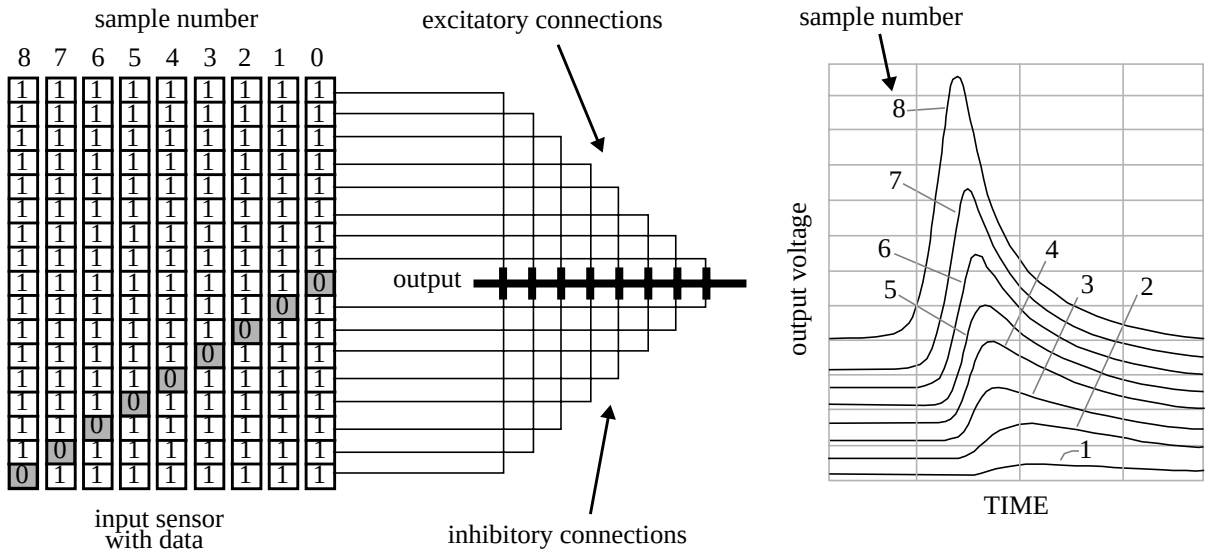

Figure 5. Input sensor with seventeen elements connected to a branch of an artificial dendritic tree which responds to the position of a target in the sensor array. The connection pattern shown, produces a dynamic response that depends nonlinearly on how far off the target is from the center. When the target is centered, the output is a null. As the target moves off center, the resultant voltage increases nonlinearly with separation distance between target and center. The sensor, with its data field, is shown at nine different times. Each sample time shows the target (in this case a 0) going off center and the resultant output (offset for clarity) from the artificial dendritic branch. A total of sixteen artificial synapses are on this branch. The top half of the sensor array connects to only excitatory artificial synapses. The bottom half connects only to inhibitory synapses. The resultant voltage transient is measured at the left end of the branch (labeled as output).

The connection patterns which represent solutions to this particular problem illustrate one of the characteristics of our dynamic neural networks: the connection pattern may form a geometric shape that reflects the spatiotemporal processing capability of the network. This is shown in figure

6 where only the connections are drawn for several possible solutions to the target tracking network. Here we see that the geometric shapes that represent solutions have a triangular form. Excitatory connections are indicated by dashed lines and inhibitory by solid lines. Figure 6a shows the connection pattern for the network of figure 5 in which adjacent synapses are connected to adjacent sensor elements. In this example, there are no gaps of unconnected artificial synapses between connected synapses. This connection pattern represents one end of the solution space for this problem. For other solutions, there may be gaps or center sensor elements may push out their connections to more distal artificial synapses like those shown in figures 6b and 6c. A plot of the expected peak output voltage for each of these solutions is shown beneath its respective connection pattern.

The connection patterns shown in figure 6 represent optimal solutions because the output of the network is a monotonic function of the target position. This can be seen in the connection pattern by observing that each sensor element connects to a more distal artificial synapse than its immediate away-from-center neighbor does. Non-optimal connection patterns result in nonmonotonic outputs which either have flat regions or regions where the output voltage changes direction. The latter type of connection pattern would be unacceptable for control applications. For the target tracking application, our goal was to find a connection pattern that closely matched the desired response and was also an optimal solution.

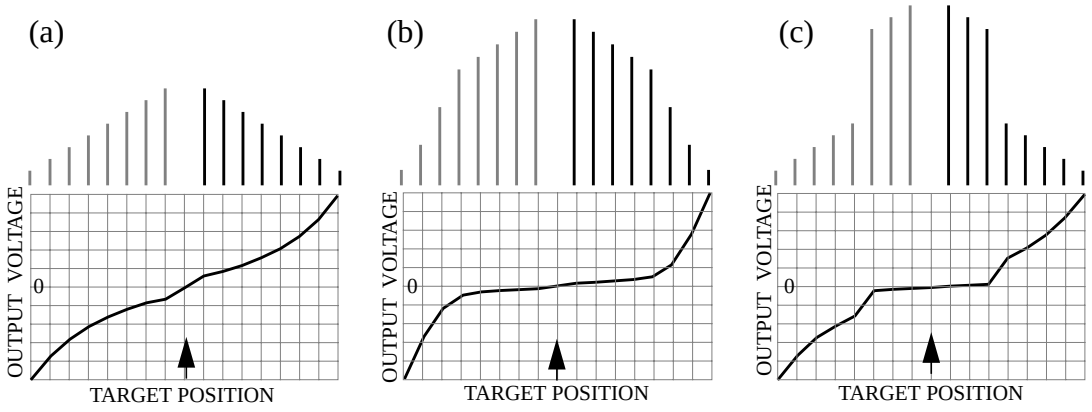

Figure 6: Three different optimal solutions to the target following problem. The peak output voltage is measured at the output node of the dendritic branch shown in figure 5 and plotted versus the position of the target in the sensor array. The center position of the sensor array is marked by an arrow. When the target is on center the output voltage should be zero. Different output voltage behavior is obtained by moving connections to either more distal or to more proximal artificial synaptic sites on the branch.

Experiments were conducted for sensor arrays having seven, nine, and seventeen elements. Each experiment continued with generation after generation of reproduction until diversity was judged to be low enough to warrant invoking the optimizer. The threshold for accepting an offspring to replace a below-average member of the population was set at the fitness value average. The annealing schedule reduced the temperature by 0.9 at each iteration of the optimizer.

The adjustable parameters for each experiment were starting temperature, population size, and number of artificial synapses. The Connection Lists were initialized before the start of each experiment with random connection patterns.

Figure 7 shows some of the experimental results for a seventeen element sensor array. Figure 7b displays the desired branch voltage response that was used as the training set. Its connection pattern is shown above the plot. Figures 7a and 7c were results obtained with the genetic algorithm when using the output response shown in figure 7b as the training set. The connection pattern in figure 7a represents a nearly optimal solution because the response is monotonic except for one point near the center. This result was obtained with a population size of 500. Notice that this solution has an extra excitatory connection near the center at sensor element 10, yet the network still manages to match the desired output fairly well. Apparently, the network compensates for the excitatory connection by having a stronger than normal inhibitory connection at sensor element 13. Figure 7c shows the results of using a population size of 100 Connection List strings. This connection pattern is not very useful because of the voltage trajectory reversal near the sensor ends.

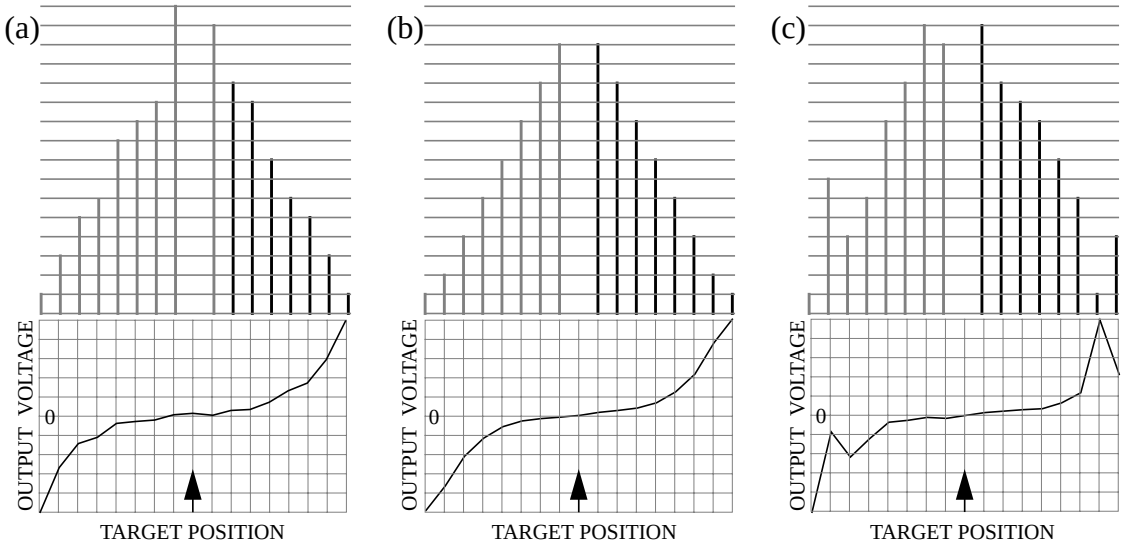

Figure 7: Results of training artificial dendritic branch having 16 excitatory and 16 inhibitory synapses for a target following application like that shown in figure 5. The horizontal lines indicate synaptic site locations. Connections at top are those to weakest artificial synapses (i.e. the most distal locations). Dashed lines indicate excitatory connections. Arrow indicates center of sensor array. Output voltage was measured at node labeled output shown in figure 5.

The results for the seven element sensor experiments were always optimal connection patterns provided that the population size was larger than 300. Although the resultant connection patterns were optimal, they did not always match the desired connection pattern exactly. The results for the nine element sensor experiments showed similar behavior in most cases.

Figure 8 shows how the best and average fitness values typically change with each generation. These results were obtained using a population size of 2000 Connection Lists, a 17 element sensor, and a single artificial dendritic branch with 32 artificial synapses. The Connection Lists are all initialized with random connections at the start of the search. The best fitness value for each generation is shown as the solid line, and the average fitness is represented by the dotted line. Several interesting behaviors can be seen in these data. The most important behavior is the rapid improvement in the best fitness value during the first 200 generations. The improvement is primarily due to crossover, which occurs in three distinct steps. The first rapid rise in the best fitness value occurs during the first one hundred generations. At about 90 generations, the average fitness value is nearly equal to the best fitness value, which implies that all of the Connection Lists are virtually the same and thus diversity is quite low. This condition renders the crossover operator ineffective.

When diversity is low, as indicated by a low crossover rate, new schemata are introduced by the embedded simulated annealing optimizer. The immediate effect of invoking this optimizer can be seen near generation 100 in figure 8, where the average fitness value, which had been steadily improving with each generation, is suddenly reduced to a level nearly equal to its initialized value. This is followed by a period of high crossover activity and a rapid improvement in the average fitness value, but little change in the best fitness value occurs until twenty generations have passed and there is once again a rapid improvement in the best fitness value. This behavior is repeated near generation 150, and after generation 200 essentially no improvement in the best fitness value is realized.

The simulated annealer clearly introduces diversity into the population of Connection Lists as shown in figure 8. The level of diversity is controlled by the temperature: higher temperatures allow more diverse Connection Lists to be added to the population. As the temperature is lowered, the extent of diversity introduced into the population is reduced. This can be seen in figure 8, where the level of the average fitness value immediately after invoking the simulated annealer rises with each application, in which the temperature is reduced. The results were very different in experiments where the simulated annealer was replaced by a randomizer that initialized all Connection Lists when diversity was low. In these experiments, the improvement in the best fitness value took many more generations to achieve the same level as with the simulated annealer. For this particular example, 300 additional generations were needed to reach the maximum fitness value.

Figure 9 shows the evolution of the best fitness value for experiments using different population sizes. The rate of convergence and the final fitness value increases with population size, as expected. However, when we compare the results for experiments having 2000 and 4000 Connection Lists we see that the experiment with 4000 strings actually takes longer to converge than the experiment with 2000 strings. It can be seen in figure 9 that around generation 180 the experiment with the larger population takes an unexpected turn and does not recover for some 200 generations.

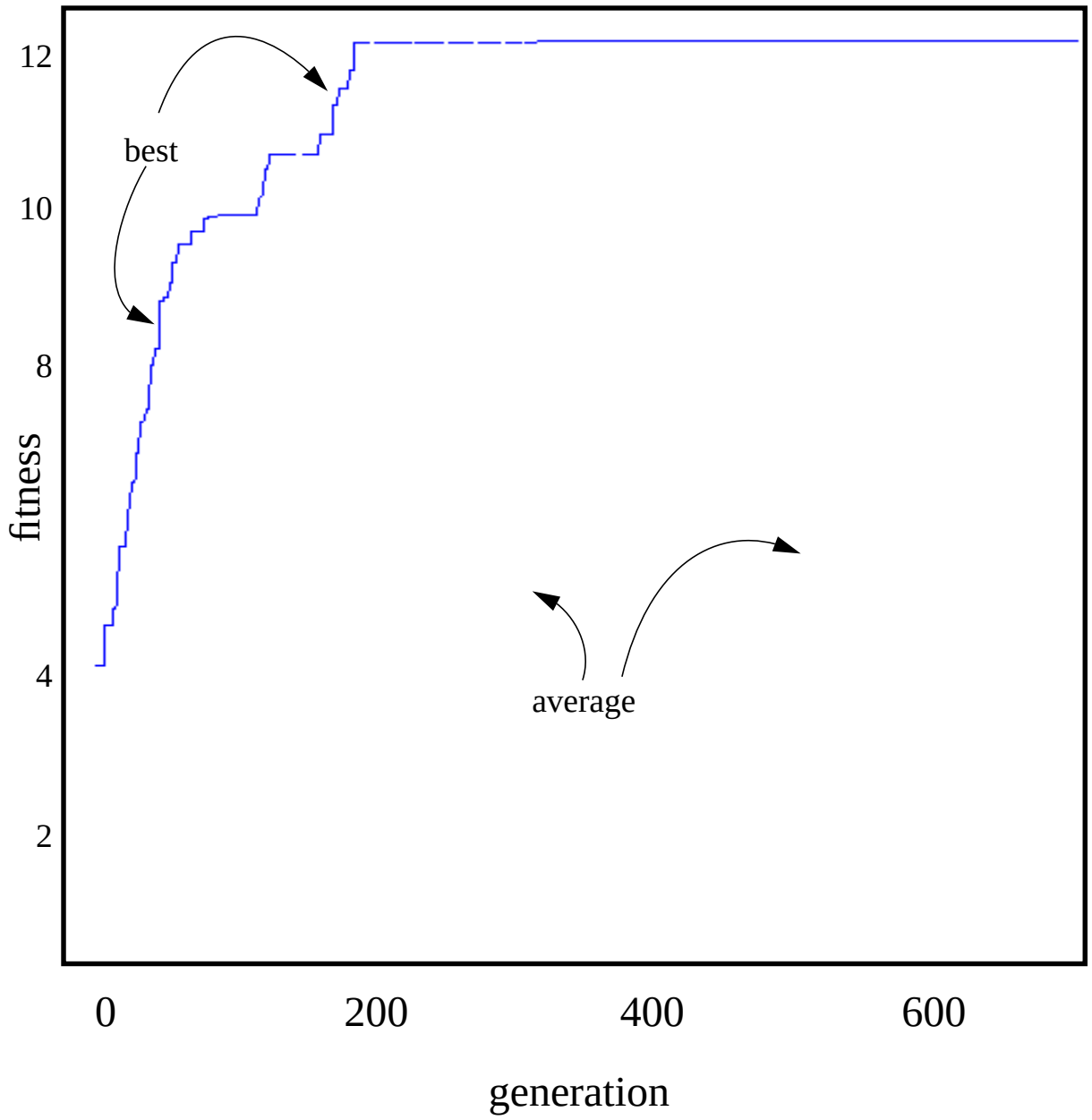

Figure 8: An example of how the best and average fitness values typically change with each generation during training for the one-dimensional target tracker. The fitness was These results were obtained using a population size of 2000 Connection Lists, a 17 element sensor, and a single artificial dendritic branch with 32 artificial synapses. The Connection Lists are all initialized with random connections at the start of the search. The best fitness value for each generation is shown as the solid line, and the average fitness is represented by the dotted line. The embedded simulated annealer is invoked through generation 420, after which a randomizer is used.

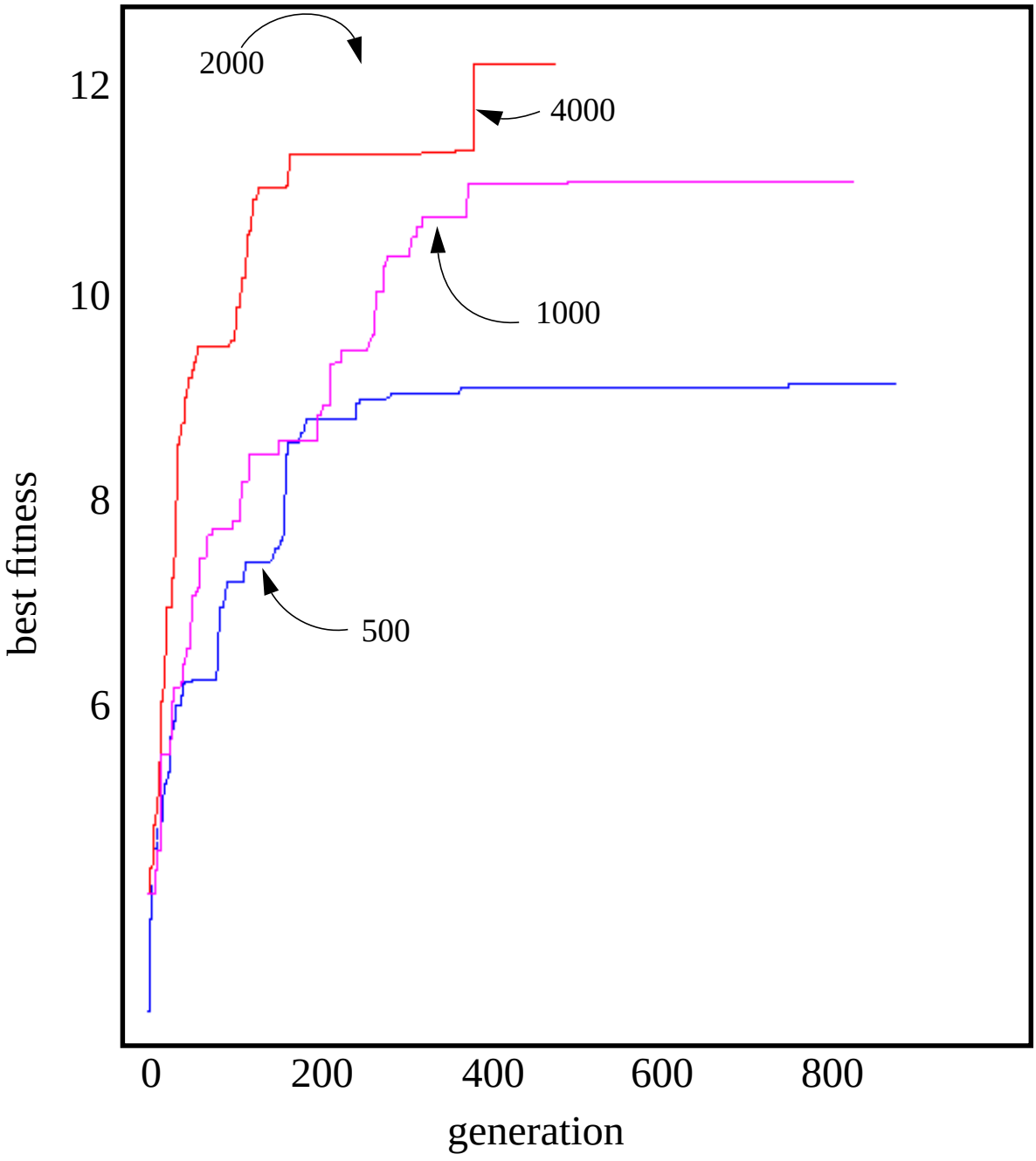

Figure 9: The evolution of the best fitness value for experiments using different population sizes. These results were obtained using a 17 element sensor and a single artificial dendritic branch with 32 artificial synapses. The Connection Lists are all initialized with random connections at the start of the search.

4 Conclusions and Future Directions

In this paper, we discussed the approach we are taking in training a new type of analog artificial neural network using a specialized genetic algorithm. The analog network is built from artificial dendritic trees which are sensitive to temporal and spatial signal characteristics and where weights are replaced by physical connections. A basic genetic algorithm was modified and used to train a simple network for a target tracking application. We found that uniform selection of mates offers a reduction in computational cost for this particular application. However, this result will have to be tested with other applications before a complete picture can be determined. We found that the crossover operator worked well on connections rather than bit strings and that an embedded optimizer in place of the mutation operator improved training performance.

Much remains to be determined before we can state that an effective and efficient genetic algorithm has been developed for our dynamic neural networks. We have described work-in-progress that leaves many questions unanswered. Most of these questions are concerned with improving the search procedure with a better understanding of the effects of population size, acceptance threshold, annealing schedule for the optimizer, and population diversity. Evaluation of different embedded optimizers is also needed. In addition, synaptic convergence and sensor divergence are often important characteristics of connection patterns, although not for the application reported here. A better understanding of how synaptic convergence and sensor/neuron divergence affect training speed is needed. Connections to artificial synapses represent utilization of valuable system resources. Therefore, an important goal of any training method should be to minimize the number of connections, but not at the expense of realizing an optimum solution to the problem. Synaptic convergence can result in an effective increase in the number of artificial synapses and should be rewarded by the fitness function. Sensor or neuron divergence tend to use up artificial synapses and should be controlled by an appropriate fitness value reduction.

Our interests lie in the direction of control applications using image data, which typically involves large numbers of sensor elements and large numbers of artificial synapses. Therefore, a better understanding of how training effectiveness changes with network size is a critical research direction for us. In addition, inclusion of past and future desired network output in the evaluation of the fitness function may prove effective in dynamic applications. And finally, since our networks are best realized in hardware we need to develop genetic algorithm techniques that are not only compatible with the hardware implementation but also take advantage of hardware capabilities.

References

[1] Rosenblatt, F. (1962) *Principles of Neurodynamics*. Washington D.C., Spartan Books

[2] Hopfield, J. J. (1982). “Neural networks and physical systems with emergent collective computational abilities,” *Proc. Natl. Acad. Sci*., 79, 2554-2558

[3] Elias, J. G. (1992). “Spatiotemporal properties of artificial dendritic trees,” *Proceedings of the IJCNN, Baltimore,* vol. II, 19-26.

[4] Elias, J. G., Chu, H. H., and Meshreki, S. (1992) “Silicon implementation of an artificial dendritic tree,” *Proceedings of the IJCNN, Baltimore*, vol. I, 154-159.

[5] Elias, J. G. and Chang, B. (1992) “A genetic algorithm for training networks with artificial dendritic trees,” *Proceedings of the IJCNN, Baltimore*, vol. I, 652-657

[6] Berry, M. and Bradley, P. (1976) “The growth of the dendritic trees of purkinje cells in the cerebellum of the rat,” *Brain Res*., 112, 1-35

[7] Whitley, D., Starkweather, T., and Bogart, C. (1990) “Genetic algorithms and neural networks: optimizing connections and connectivity,” *Parallel Computing* 14, 347-361

[8] Koza, J. R. and Rice, J. P. (1991) “Genetic generation of both the weights and architecture for a neural network,” *Proceedings of the IJCNN, Seattle*, vol. II, 397-404

[9] Brent, R. P. (1973) *Algorithms for Minimization without Derivatives*, Englewood Cliffs, NJ Prentice-Hall

[10] Montana, D. and Davis, L. (1989) “Training feedforward neural networks using genetic algorithms,” *Proc. Internat. Joint Conf. Artificial Intelligence*, 762-767

[11] Ackley, D. H. (1987) in *Genetic Algorithms and Simulated Annealing*, L. Davis, Ed., Morgan Kaufmann, chap. 13

[12] Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems.* Ann Arbor, University of Michigan Press

[13] Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P. (1983) “Optimization by simulated annealing,” *Science*, 220, 671-680