Autonomous self-configuration of artificial neural networks for data classification or system control

Wolfgang Fink

Wolfgang Fink, "Autonomous self-configuration of artificial neural networks for data classification or system control," Proc. SPIE 7331, Space Exploration Technologies II, 733105 (20 May 2009); doi: 10.1117/12.821836

SPIE.

Event: SPIE Defense, Security, and Sensing, 2009, Orlando, Florida, United States

Autonomous Self-Configuration of Artificial Neural Networks for Data Classification or System Control

Wolfgang Fink[a*]
[a]California Institute of Technology, *Visual and Autonomous Exploration Systems Research Laboratory*, Division of Physics, Mathematics & Astronomy, 1200 E California Blvd, Mail Code 103-33, Pasadena, CA 91125, USA

ABSTRACT

Artificial neural networks (ANNs) are powerful methods for the classification of multi-dimensional data as well as for the control of dynamic systems. In general terms, ANNs consist of neurons that are, e.g., arranged in layers and interconnected by real-valued or binary neural couplings or weights. ANNs try mimicking the processing taking place in biological brains. The classification and generalization capabilities of ANNs are given by the interconnection architecture and the coupling strengths. To perform a certain classification or control task with a particular ANN architecture (i.e., number of neurons, number of layers, etc.), the inter-neuron couplings and their accordant coupling strengths must be determined (1) either by a priori design (i.e., manually) or (2) using training algorithms such as error back-propagation. The more complex the classification or control task, the less obvious it is how to determine an a priori design of an ANN, and, as a consequence, the architecture choice becomes somewhat arbitrary. Furthermore, rather than being able to determine for a given architecture directly the corresponding coupling strengths necessary to perform the classification or control task, these have to be obtained/learned through training of the ANN on test data. We report on the use of a Stochastic Optimization Framework (SOF; Fink, SPIE 2008) for the autonomous self-configuration of Artificial Neural Networks (i.e., the determination of number of hidden layers, number of neurons per hidden layer, interconnections between neurons, and respective coupling strengths) for performing classification or control tasks. This may provide an approach towards cognizant and self-adapting computing architectures and systems.

Keywords: Artificial neural networks, network architecture, neural couplings, coupling strengths, neurons, stochastic optimization framework, autonomous self-configuration, simulated annealing, network architecture, robustness, training, data classification, system control

1. INTRODUCTION

Artificial neural networks (ANN), such as multi-layered feedforward networks (e.g., multi-layered perceptrons), multi-layered recurrent networks, and fully connected attractor networks (e.g., Hopfield attractor networks), are at the core of Artificial Intelligence (AI) and Cognizant Computing Systems [1, 2]. ANNs are powerful methods, most prominently for:

(a) The classification and analysis of multi-dimensional data

(b) The learning of rules underlying data (i.e., so-called "generalization")

(c) The control of dynamic, highly non-linear systems (e.g., autopilots [3]).

In general terms, ANNs consist of mathematical/computational neurons (e.g., McCulloch-Pitts neurons [4]) that are binary or real-valued entities combined with sigmoidal transfer functions, such as tanh(x), to imitate the action potentials in biological neurons. These neurons are assembled in layers in the case of feedforward networks [1, 2], which are interconnected by real-valued or binary neural couplings or weights, which act as inputs to or outputs from respective neurons for the propagation of information (Fig. 1). In the case of attractor networks, such as Hopfield networks (Fig. 2) [5, 1, 2], the neurons are fully interconnected as a non-layered ensemble, acting both as input and output neurons that undergo a dynamic iteration process to update their states. ANNs try imitating the processing of biological brains. Hence

* e-mail: wfink@autonomy.caltech.edu; phone: +1-626-395-4587; website: http://autonomy.caltech.edu

Space Exploration Technologies II, edited by Wolfgang Fink, Proc. of SPIE Vol. 7331, 733105 · © 2009 SPIE · CCC code: 0277-786X/09/$18 · doi: 10.1117/12.821836

they are widely used for learning algorithms, knowledge creation, and as an essential element in the quest for cognizant computing architectures and systems.

The classification and generalization capabilities of ANNs arise from the interconnection architecture and the coupling strengths. To perform a certain classification or control task with a particular ANN architecture (i.e., number of neurons, number of layers, etc.), the inter-neuron couplings and their accordant coupling strengths have to be determined (1) either by a priori design, or (2) by using training algorithms such as Error Back-Propagation [6, 7]. However, the more complex the classification or control task, the less obvious it is how to determine an a priori design of an ANN, and, as a consequence, the architecture choice becomes arbitrary, inefficient, or altogether impossible to determine. Furthermore, rather than being able to determine directly for a given architecture the corresponding coupling strengths necessary to perform the classification or control task at hand, these have to be obtained/learned through extremely time-consuming training sessions of the ANN on test data.

Therefore, to overcome this dilemma and to pave the way towards cognizant computing architectures and systems, we are employing a *Stochastic Optimization Framework* [8] together with rapidly converging and readily parallelizable Simulated Annealing related algorithms [9, 10, 8]. For user-defined tasks (e.g., data classification, prediction, or system control) this Stochastic Optimization Framework, in conjunction with Simulated Annealing as its Optimization Engine, has the potential for:

(1) Autonomously evolving suitable ANN architectures from scratch (i.e., determine number of hidden layers, number of neurons per layer, and neural interconnectivity; Fig. 1)

(2) Determining sets of (binary or real-valued) coupling strengths for these ANN architectures for a successful performance outcome.

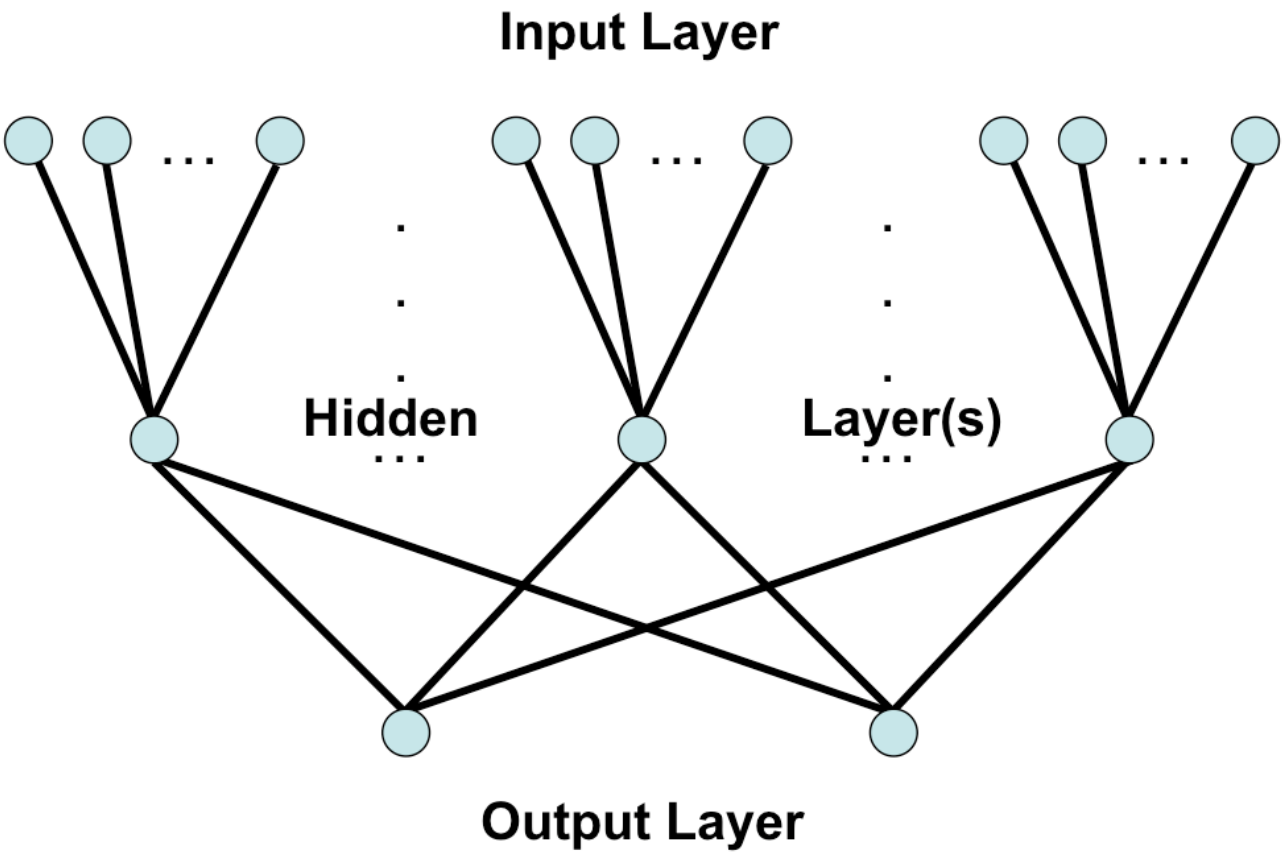

Fig. 1. Treelike feedforward multi-layer perceptron with neural input, hidden, and output layers.

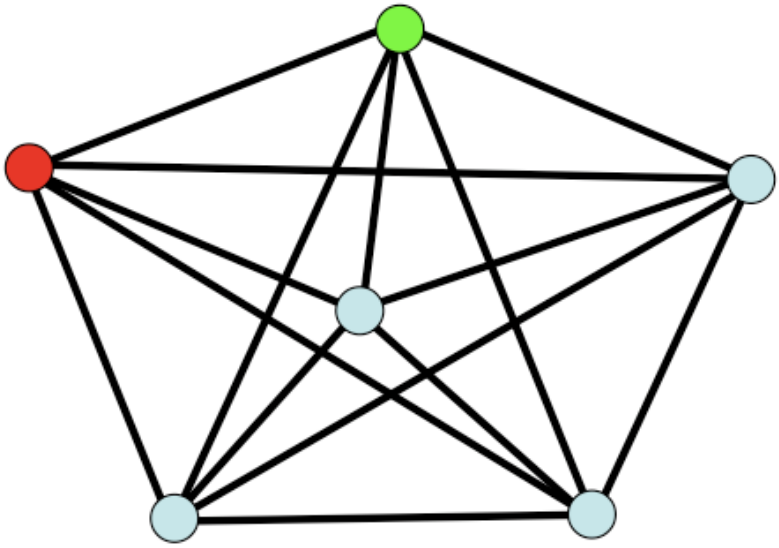

Fig. 2. Fully connected Hopfield attractor network.

2. METHODS AND TECHNICAL IMPLEMENTATION

A *Stochastic Optimization Framework (SOF, Fig. 3)*, introduced by Fink in 2008 [8], allows for efficient sampling of the entire ANN-intrinsic coupling space by repeatedly executing the given classification or control task by the SOF-configured ANN, and by comparing the outcomes against the desired outcome (e.g., low to no classification or control error), which results in a fitness measure. The goal of the SOF is to optimize this fitness.

The following two subsections 2.1 and 2.2 are directly cited from Fink, 2008 [8]:

2.1 "Stochastic Optimization Framework (SOF)

A Stochastic Optimization Framework (SOF, Fig. 3) allows for efficient sampling of the entire model-intrinsic parameter space by repeatedly running the respective model forward (e.g., on a single, cluster, or parallel computer) and by comparing the outcomes against the desired outcome, which results in a fitness measure. The goal of the SOF is to optimize this fitness. This approach is in sharp contrast to optimizing around a point design, which is often the case in engineering. Deterministic optimization techniques, such as gradient-based steepest-descent methods, are powerful and efficient in problems that exhibit only few local minima in the solution space. However, when dealing with multiple or infinite numbers of local minima, heuristic stochastic optimization methods, such as Simulated Annealing [9, 10] related algorithms, Genetic Algorithms [11, 12], other Evolutionary Algorithms, and Genetic Programming [13], may become the prime methods of choice because of their capability to overcome local minima (Fig. 4). In our case we choose modified Simulated Annealing algorithms as the optimization engine for SOFs.

2.2 Optimization Engine – Simulated Annealing (SA)

Simulated Annealing (SA) [9, 10] is a widely used and well-established optimization technique, especially for high-dimensional configuration spaces. The goal is to minimize an energy/fitness function E, which is a function of N variables or parameters. The iterative minimization process is performed by randomly changing the values of one or more of the N variables within their respective, defined value ranges, and by subsequently reevaluating the energy function E per iteration step. Two cases can occur:

(1) The change in the variable values results in a new, lower energy function value;

(2) The energy function value is higher or unchanged.

In the first scenario the new set of variable values is stored and the change accepted. In the second scenario, the new set of variable values is only stored with a certain likelihood (Boltzmann probability, defined by an annealing temperature).

This ensures that the overall optimization process does not get trapped in local minima too easily such as is the case with, for example, gradient-based, steepest-descent ("greedy") downhill optimization (Fig. 4). The annealing temperature directly influences the Boltzmann probability by making it less likely to accept an energetically unfavorable step, the longer the optimization lasts (also known as the "cooling schedule"). Then the overall procedure is repeated until the annealing temperature has reached its end value, or a preset number of iterations has been exceeded, or the energy function E has reached an acceptable user-defined level."

In contrast to *Genetic Algorithms (GA)* [11, 12] and other population-based *Evolutionary Algorithms (EA)*, SA is not population-based. Furthermore, SA is also characterized by very few user-defined parameters, mostly pertaining to the cooling schedule, making it much more suitable to tackle the actual optimization problem rather than the intricacies of the optimization algorithm used [8]. Another major advantage of SA is its "embarrassingly" parallel nature: Each available CPU can host an independent SA run without any message passing as opposed to population-based GAs or other EAs that require, at the very least, the passing of fitness evaluation information. As a result, an almost perfect linear speedup can be expected (with the exception of start-up time), which we have successfully demonstrated on a 1,024 CPU cluster computer [8].

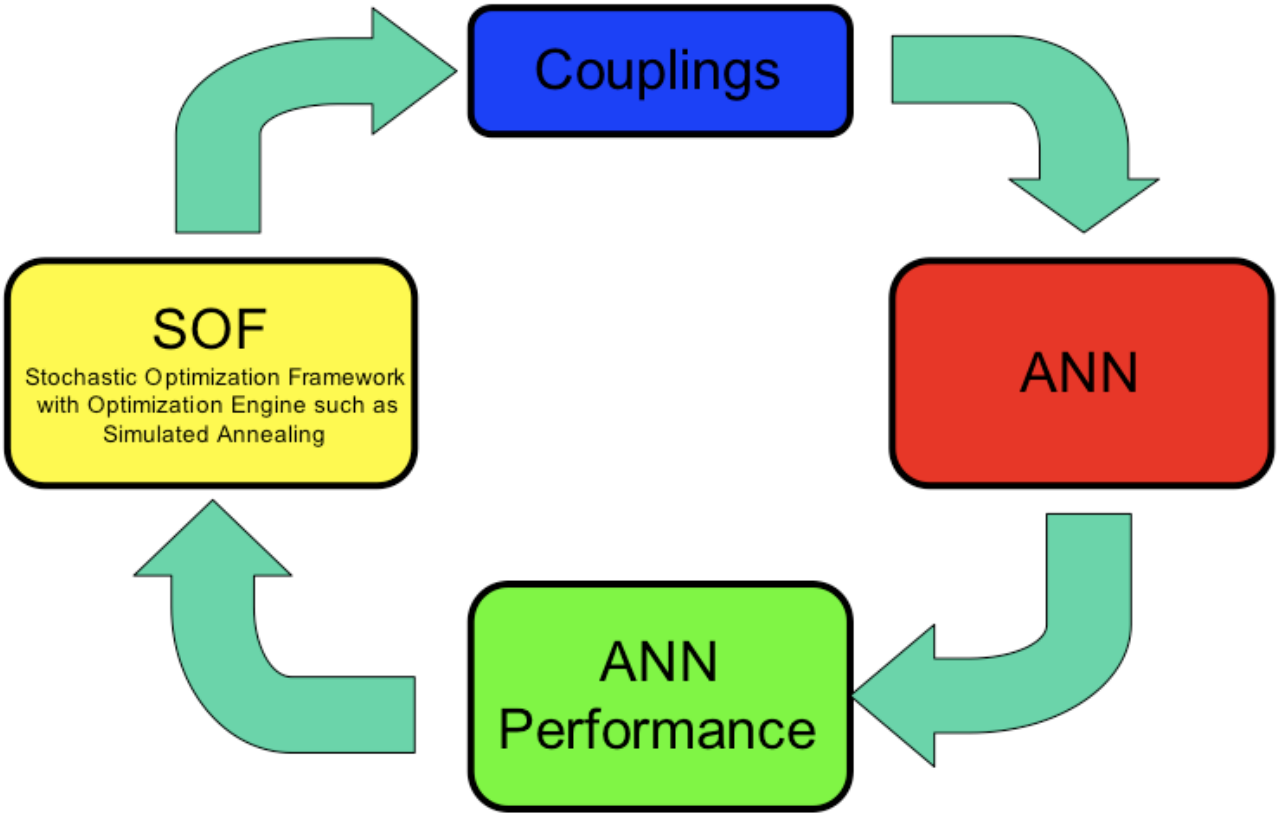

Fig. 3. Functional schematic of a Stochastic Optimization Framework (SOF) applied to autonomous Artificial Neural Network (ANN) self-configuration: The SOF efficiently samples the entire ANN-intrinsic parameter space (i.e., neural architecture and coupling strengths) by repeatedly running the current ANN on a test data set or control task (e.g., on a single, cluster, or parallel computer) and by comparing the (classification/generalization/control) outcomes against a desired outcome, which results in a fitness measure. The goal of the SOF is to optimize this fitness by using (in our case) modified Simulated Annealing algorithms as the optimization engine.

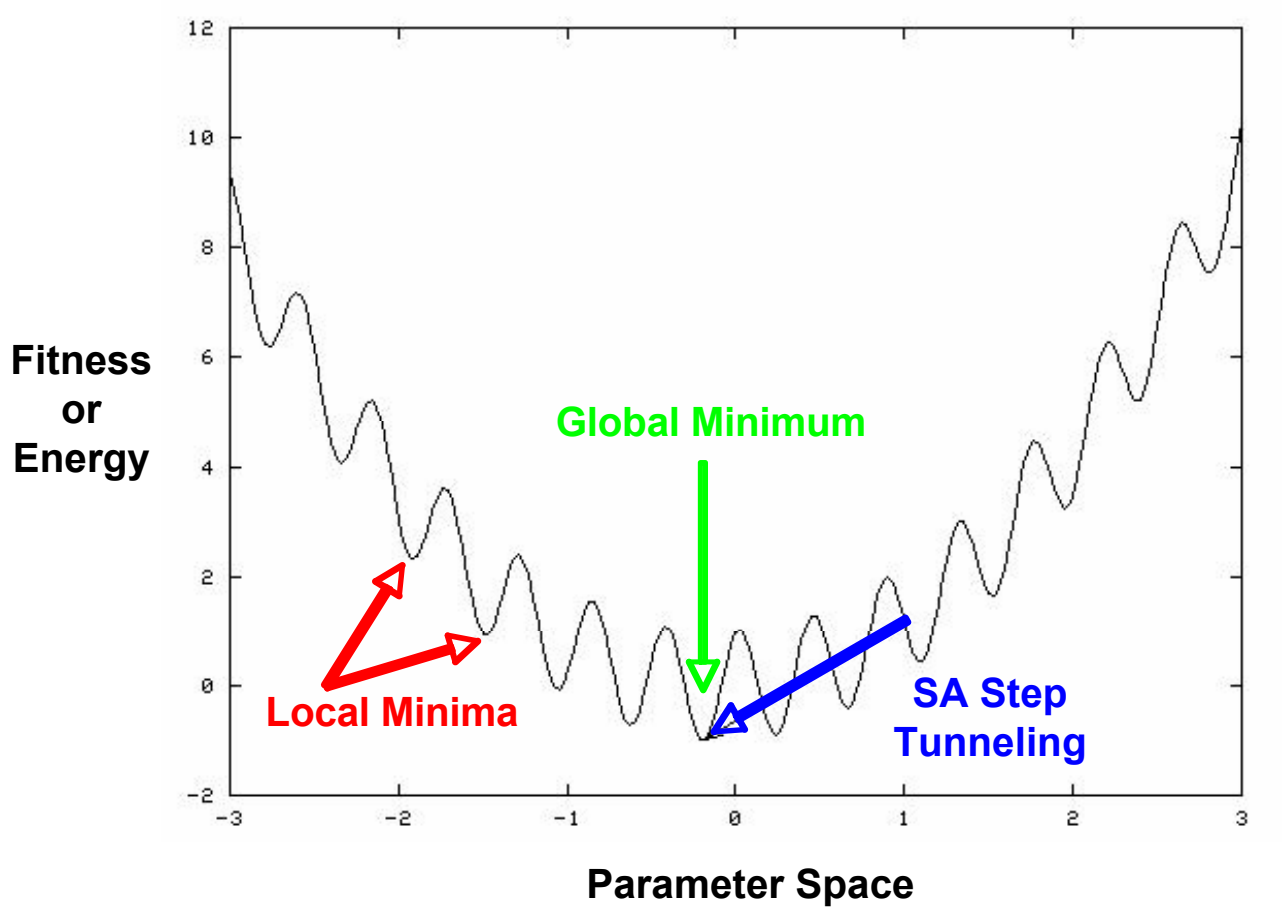

Fig. 4. Example of a fitness or energy landscape with multiple local minima. Deterministic, gradient-based, steepest-descent ("greedy") downhill optimization algorithms tend to get trapped in local minima without ever reaching the global minimum. Heuristic, stochastic optimization algorithms, such as Simulated Annealing [9, 10], Genetic Algorithms [11, 12], and other Evolutionary Algorithms possess mechanisms to "tunnel" through fitness/energy barriers to reach the global minimum, or at least to not get trapped in local minima for too long (from [8]).

2.3 Setup for Autonomous SOF-based ANN Architecture Design

In the example case of a feedfoward network, the only fixed/specific parameters for an ANN architecture design are the numbers of input and output neurons. The most general and flexible case of an SOF-based ANN design would be realized by keeping the number of hidden layers and the respective number of hidden neurons per hidden layer completely undefined and subjected to the optimization process. This can be implemented using, for example, linked lists of neurons and respective couplings. The downside of this approach is a severely prolonged optimization time (i.e., number of SOF iterations) necessary to achieve a given classification or control task. A computationally more feasible approach can be realized by predefining/overdefining the problem: using a larger number of hidden layers with respective hidden units per layer than would be expected to solve the given classification or control task. The SOF-based iteration process would subsequently "thin" out this overdefined ANN architecture to its leanest possible (i.e., minimal) architecture that successfully performs (i.e., no or low classification/control error) the given classification/control task. This "leaning" process can be further enforced by modifying the fitness function accordingly, for example, by adding the number of active neural couplings per architecture. A minimization of the fitness function would thereby also try to reduce the number of active neural couplings and hence thin out the ANN architecture. However, a balance must be struck such that the leaning process does not impose negatively on the classification/control capability of the ANN.

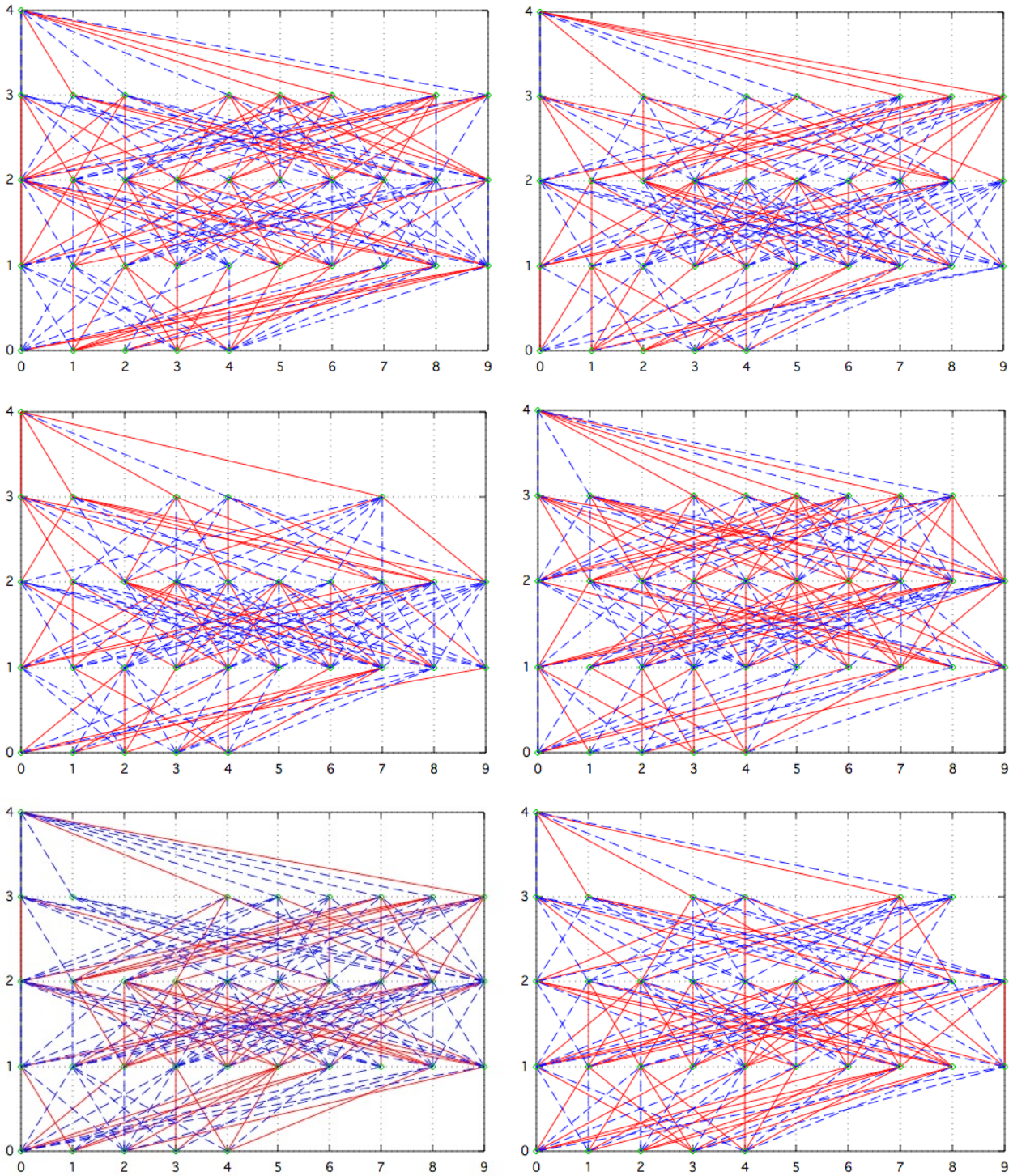

Fig. 5. Six snapshots of the SOF-based automated design of a five layer feedforward neural network with four input units (layer 0, bottom), three hidden layers with 10 hidden units each (layers 1-3), and one output unit (layer 4, top). The network uses binary couplings with red = +1 and blue = -1.

Figure 5 shows a sequence of a neural network architecture development driven by the SOF. Displayed is a network with four input units, three hidden layers with 10 hidden units each, and one output unit. Figure 5 demonstrates how the SOF-based design operates on the architecture and the neural couplings respectively by creating/deleting couplings, changing coupling strengths, and eliminating hidden units if there are no incoming and outgoing neural couplings (or recreating them if there are).

3. RESULTS

In the following we present an instructive example application to illustrate the overarching concept of applying a Stochastic Optimization Framework to the autonomous design of Artificial Neural Networks architectures, including the determination of the neural coupling strengths. We used a modified Simulated Annealing algorithm as the Optimization Engine for the SOF [8].

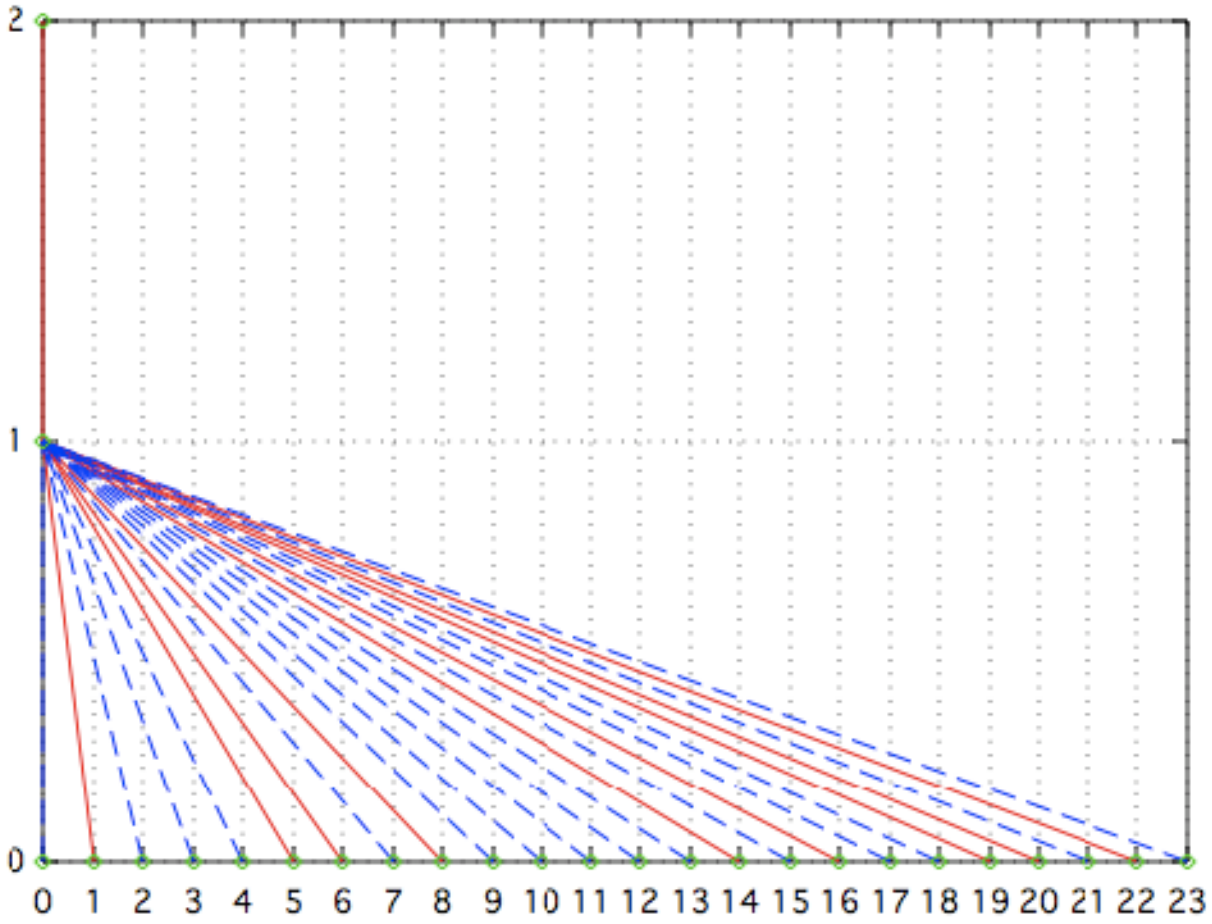

Fig. 6. Start condition for the 24-bit even/odd number classifier: a three layer feedforward neural network, consisting of the input layer with 24 neurons (corresponding to the 24 bits), one hidden layer with one hidden neuron, and one output layer with a single output neuron. For the neural couplings we considered binary couplings ±1. The start condition was a fully connected network with a random choice of binary coupling strengths (red = +1 and blue = -1).

3.1 24-Bit Even/Odd Number Classifier

To determine whether a 24-bit number is even or odd, we used a three layer feedforward neural network, consisting of the input layer with 24 neurons (corresponding to the 24 bits), one hidden layer with one hidden neuron, and one output layer with a single output neuron. For the neural couplings we considered binary couplings ±1. The start condition was a fully connected network with a random choice of binary coupling strengths (Figure 6). The training set used for the fitness evaluation consisted of 10,000 randomly drawn numbers out of a possible 16,777,216 numbers (24-bit integer).

After 336 optimization iterations via the SOF the neural network design automatically came up with the expected architecture and according neural coupling strengths: preserving and using only the first bit input neuron coupling to the hidden unit, and the coupling from the hidden unit to the output unit (Figure 7). Both binary couplings have the coupling strength of -1 (note: +1 would have been an equivalent alternative solution).

This instructive example shows that the SOF-mediated optimization process eliminated 23 unnecessary input neurons, and it also changed the remaining two couplings to equal coupling strength (note that in the start configuration the coupling strengths from the first bit input neuron to the hidden unit and from the hidden unit to the output unit are different).

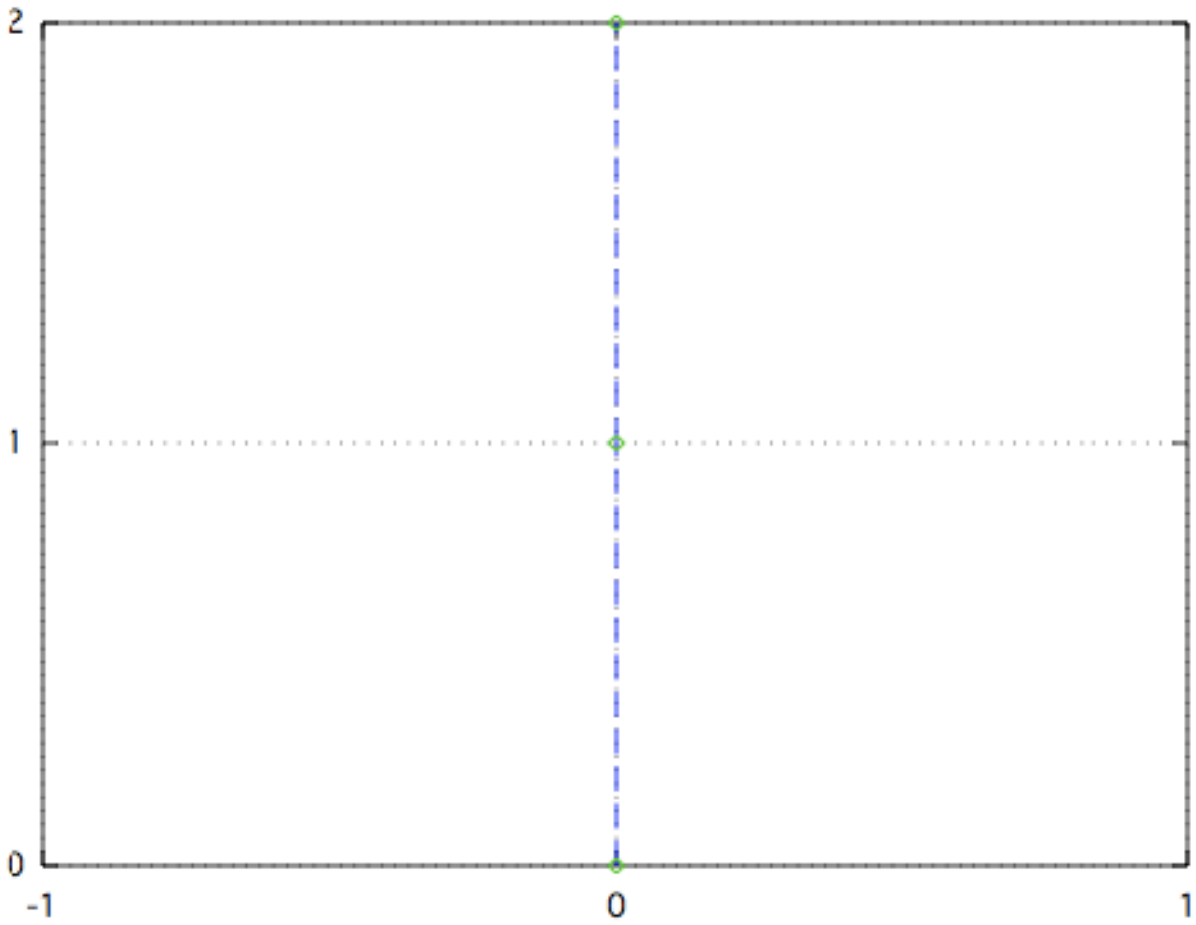

Fig. 7. Final 24-bit even/odd number classifier after 336 optimization iterations via the SOF. The neural network design automatically arrived at the expected architecture: preserving and using only the first bit input neuron coupling to the hidden unit, and the coupling from the hidden unit to the output unit. Furthermore, both couplings are the same coupling strength of -1 (blue).

Figure 8 shows the classification error of the respective SOF-mediated ANN architecture on the training set as a function of SOF iterations.

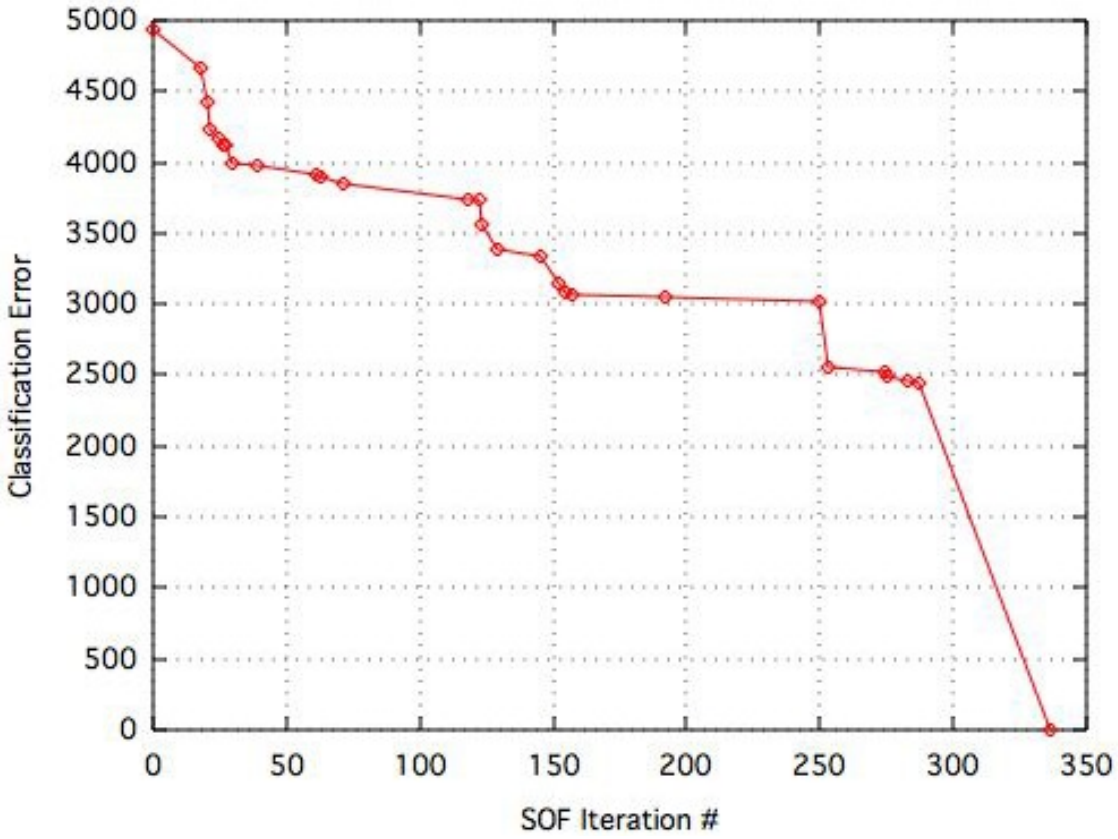

Fig. 8. Classification error of the respective SOF-mediated ANN architecture on the training set as a function of SOF iterations.

4. DISCUSSION & OUTLOOK

Stochastic Optimization Frameworks (SOF) in conjunction with efficient optimization engines, such as Simulated Annealing, are powerful in that they can routinely match or exceed human expert performance in the optimization of processes, system design and performance [8]. Deployed ANNs, whether in software or hardware (e.g., using neural processing chips), e.g., aboard a spacecraft for onboard data processing or spacecraft control, may be subjected to harsh environments such as the radiation environment around Europa. This may lead to the corruption of neural couplings, thus degrading or annihilating the classification or control performance of the ANN used. The proposed SOF-based autonomous self-configuration of artificial neural networks may have the potential to overcome or mitigate these adverse effects through rapid retraining, thereby increasing fault-tolerance and robustness.

Furthermore, the proposed approach of SOF-based ANN design reduces the arbitrariness of neural network architectures for given classification or control tasks due to the underlying optimization process. It removes the need for the a prior design of an ANN as well as the subsequent training of the ANN on test data via general or specialized training algorithms (such as Error Back-Propagation [6, 7]) to determine the appropriate neural coupling strengths. As such SOF-based ANN design may provide an approach towards cognizant and self-adapting computing architectures and systems.

REFERENCES

1. Hertz J, Krogh A, Palmer RG, *Introduction To The Theory Of Neural Computation*, Lecture Notes Volume I, Addison-Wesley Publishing Company, 1991.
2. Müller B, Reinhardt J, *Neural Networks: An Introduction*, Springer, Berlin Heidelberg New York, 1990.
3. Bassi D, Fink W, Optimal Attitude Control Parameters Via Stochastic Optimization Framework for Autonomous Aircraft; *IEEE Aerospace Conference Proceedings*, paper #1753, Big Sky, Montana, 2009.
4. McCulloch W, Pitts W, A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, 7:115-133, 1943.
5. Hopfield JJ, Neural networks and physical systems with emergent collective computational abilities *Proc. Natl. Acad. Sci. USA* **79** 2554–8, 1982.
6. Rumelhart DE, Hinton GE, Williams RJ, Learning representations by back-propagating errors Nature **323** 533–6, 1986.
7. Rumelhart DE, Hinton GE, Williams RJ, Learning internal representations by error propagation Parallel Distributed Processing ed D E Rumelhart and J L McClelland (Cambridge: MIT Press) 1986.
8. Fink W, Stochastic Optimization Framework (SOF) for Computer-Optimized Design, Engineering, and Performance of Multi-Dimensional Systems and Processes; *Proc. SPIE*, Vol. 6960, 69600N (2008); DOI:10.1117/12.784440 (invited paper).
9. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E, Equation of State Calculation by Fast Computing Machines, *J. of Chem. Phys.*, 21, 1087 – 1091, 1953.
10. Kirkpatrick S, Gelat CD, Vecchi MP, Optimization by Simulated Annealing, *Science*, 220, 671 – 680, 1983.
11. Holland JH, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, Michigan, 1975.
12. Goldberg DE, Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, 1989.
13. Koza JR, *Genetic Programming: On the Programming of Computers by Means of Natural Selection,* Cambridge, MA: The MIT Press, 1992.