Salzburg Interest Group on Integrated Systems

siGis-016-1998 July 3, 1998

Symbiotic Coevolution of Artificial Neural Networks and Training Data Sets

Fifth International Conference on Parallel Problem Solving from Nature
September 27–30, 1998, Amsterdam

Helmut A. Mayer

helmut@cosy.sbg.ac.at

Department of Computer Science

University of Salzburg

siGis
Salzburg Interest Group on Integrated Systems

Correspondence to:

Universität Salzburg
Institut für Computerwissenschaften und Systemanalyse
Jakob–Haringer–Straße 2
A–5020 Salzburg
Austria

Symbiotic Coevolution of Artificial Neural Networks and Training Data Sets

Helmut A. Mayer

University of Salzburg, Department of Computer Science, A–5020 Salzburg, Austria
helmut@cosy.sbg.ac.at

Abstract. Among the most important design issues to be addressed to optimize the generalization abilities of trained artificial neural networks (ANNs) are the specific architecture and the composition of the training data set (TDS). Recent work has focused on investigating each of these prerequisites separately. However, some researchers have pointed out the interacting dependencies of ANN topology and the information contained in the TDS. In order to generate coadapted ANNs and TDSs without human intervention we investigate the use of symbiotic (cooperative) coevolution. Independent populations of ANNs and TDSs are evolved by a genetic algorithm (GA), where the fitness of an ANN is equally credited to the TDS it has been trained with. The parallel netGEN system generating generalized multi–layer perceptrons being trained by error–back–propagation has been extended to coevolve TDSs. Empirical results on a simple pattern recognition problem are presented.

1 Introduction

Dealing with real world problems we often face the problem of huge amounts of possible training data patterns, e.g., a robot collecting sensor data from its environment, or satellite image pixels to be classified. In order to construct (small) TDSs the ANN can extract the most useful information – based on certain quality measures – of, a variety of *Active Selection* methods have been proposed in literature [Zha94] [Röb94] [Plu94]. Hereby, a small TDS is generated iteratively by selecting the most useful patterns out of all available data. In [MS97] we proposed *Evolutionary TDSs*, a GA–based active selection method, where the patterns of the (sub)–optimal TDSs are selected in parallel.

With all these approaches the ANN topology remains fixed during the construction of the TDS. Evidently, the choice of the ANN architecture will influence the training patterns selected for the TDS, hence, the overall performance of the network. As a consequence, the work of Zhang [Zha93] and Plutowski [Plu94] suggests to construct ANN architecture and TDS contents simultaneously. In this paper we present an evolutionary approach to this problem by coevolving ANN architectures and complete TDSs in independent populations. Thus, in extension to previous work (Section 1.2) we not only search individual useful training patterns, but a collection of cooperating patterns (TDS).

1.1 A Brief Taxonomy

With most applications of *Evolutionary Algorithms* solutions (individuals) to a problem are adapted to a static environment (fitness function). While this is certainly sufficient for a variety of real world problems having well–defined (and static) goals, certain problem domains, e.g., *Learning Robots* operating in a dynamic environment (possibly other robots), call for coevolution of individual(s) and environment(s).

The biochemist Lovelock writes in [Lov88]:

> "The evolution of a species is inseparable from the evolution of its environment. The two processes are tightly coupled as a single indivisible process."

Specifically, with the coevolution of ANNs and TDSs, the terms individual and environment can be used interchangeably, as an individual ANN (TDS) is exhibited to an environment TDS (ANN). This in mind, we will use the terms *Population* I for individuals and *Population* E for environments in the following.

Coevolution can be *Competitive (Parasitic)* or *Cooperative (Symbiotic)*, i.e., an environment (of population E) increases its fitness by decreasing or increasing the fitness of an individual in population I, respectively. The *Lifetime* of individuals in I and E can be different, e.g., E remains unchanged, while I evolves for a number of generations, or vice versa. The fitness evaluation of an individual in I can be done in a *Single* environment, m *Multiple* environments (with $m < n$, the number of environments in E), or in *All* environments n. When evaluating in multiple or all environments, the various fitness values have to be aggregated for the single individual, e.g., best or average fitness.

1.2 Related Work

Based on the pioneering work of Hillis [Hil90], Paredis [Par94] competitively coevolved the weights (evolutionary learning) of ANNs of fixed topology for a synthetic classification task. The population E consists of a fixed number of training patterns. Parasitic patterns are presented sequentially to ANNs, and the fitness of a network is based on the number of correct classifications of the last 20 examples [1]. The (parasitic) fitness of an example is given by the number of times it was incorrectly classified by ANNs it encountered most recently. In a strict sense the examples do not evolve, as they only change fitness without alteration of the patterns [Par94].

Potter and De Jong [PJ95] proposed *Cooperative Coevolutionary Algorithms* (CCGAs) for the evolution of ANNs of cascade network topology. In this approach the population I is divided into species corresponding to subtasks (ANN subnets). The quality of the ANN composed of subnets determines the fitness of

---

[1] In [Par94] this procedure is termed *Life Time Fitness Evaluation*, but it is not consistent with the definition of lifetime we gave above.

the contributing individuals of different species. Thus, cooperating individuals are rewarded inducing the evolution of collaborative species. However, with this approach the TDS never changes and represents a static environment. Coevolution occurs within population I by decomposing the problem into interacting subtasks (collaborative species).

Barbosa [Bar97] employs a competitive coevolutionary GA for structural optimization problems using a game–theoretic approach. The first player, the designer of a mechanical structure, tries to minimize the *Compliance* (a measure of the overall deformability under a specific load). The second player, nature, challenges the designer's constructions by maximizing the compliance. The choices of the two players are evolved in independent populations I (structures) and E (loads) with alternating lifetime of individuals (each population is kept "frozen" for a certain number of generations) and fitness evaluation in all environments.

2 ANN and TDS encoding

The technical platform for the coevolution of ANNs and TDSs is the netGEN system [HMS95] which has been designed for an arbitrary amount of workstations employing the *Parallel Virtual Machine (PVM)* library [GBD+94] in order to train individual ANNs in parallel.

The ANN's genetic blueprint is based on a direct encoding suggested by Miller et al. [MTH89]. With this approach each connection is represented in a binary adjacency matrix called *Miller–Matrix* (MM) describing ANN architecture. Contrary to the original crossover operator exchanging rows and columns of the MM, we use standard 2–point crossover operating on a linearized MM where the triangle matrix of fixed 0s (feed–forward architecture) is not included.

The main enhancement in our *Modified–Miller–Matrix (MMM)* direct encoding scheme [HMS95] are the *Neuron Markers* which are encoded in a separate section on the ANN chromosome (the two other sections contain learning parameters and all possible connections, respectively). Technically, each neuron marker (one bit) is transferred to the main diagonal (i, i) of the MMM adjacency matrix, indicating the absence or presence of a particular neuron i and its connections (Fig. 1). As a consequence, most ANN chromosomes contain non–coding regions [2], as all connections associated with a specific neuron become obsolete, if the corresponding neuron marker is zero. The non–coding regions (*Introns*) reduce the disruptive effects of crossover [Lev91] [Wu96] [May97].

The maximum number of hidden neurons (neuron markers) has to be set in advance with this encoding scheme, hence, it could be labeled as *Evolutionary Pruning*, since the system imposes an upper bound on the complexity of the network. One of the measures to avoid overfitting is the encoding of the number of training epochs with the additional benefit of another ANN parameter being set automatically. Two learning parameters are also encoded in the genotype.

[2] Non–coding DNA regions or *Junk DNA* can be found in abundance in biological systems, i.e. *Eukaryotes.*

The standard back–propagation learning rate η, and a parameter δ_ϵ [3] which may also reduce overfitting, as the ANN can often be trained with substantial fewer epochs.

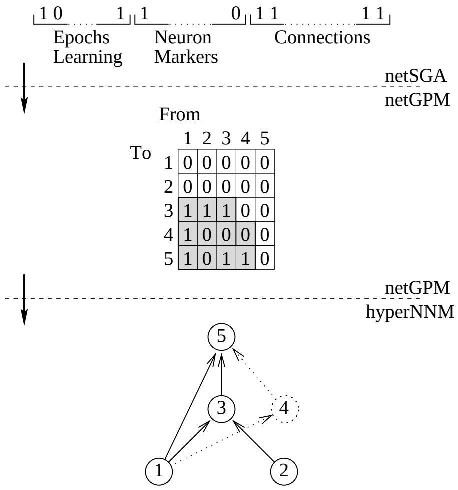

Fig. 1. ANN encoding scheme.

In addition to our modifications of the MM encoding scheme, the GA does not operate on the matrix representation, but on the concatenation of the rows of the non–zero part of the MMM's lower triangle matrix representing the possible connections. All MMM entries clamped to zero need not be processed by the GA, hence, they are not encoded in the genotype.

The TDSs are encoded as a concatenation of the desired number of training patterns (genes) which are represented as binary coded indexes to a data store file containing all available training patterns (Fig. 2). Clearly, the number of bits used to index an item in the data store is dependent on the size of the store. This encoding scheme enables the selection of one or more copies of a pattern into the TDS. A more conventional encoding for evolution of subsets [RG93] would constrain the search space which is advantageous for most set problems, however, the inclusion of multiple pattern copies might be beneficial in the case of TDSs, and should not be ruled out in advance.

Both GAs, netSGA and dataSGA, use the same basic genetic operators which are given below. For the GA selection phase we chose *Binary Tournament Selection* without replacement.

---

[3] δ_ϵ specifies an error threshold for all output neurons . If the error is below that threshold, the output neuron is considered to have zero error.

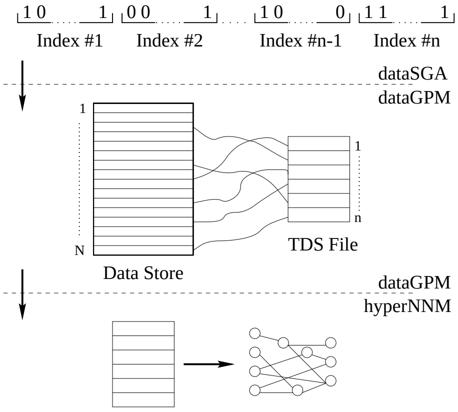

Fig. 2. TDS encoding scheme.

2.1 ANN and TDS Fitness Functions

The ANN fitness function comprises a complexity regularization term $\mathcal{E}_c$ suggested by Rumelhart [HP89]

$$\mathcal{E}_c = \sum_{i \in C_{total}} \frac{w_i^2}{1 + w_i^2}, \tag{1}$$

where w_i is the weight of a connection and C_{total} is the set of all network connections. The *Composite Fitness Function* $\mathcal{F}$ is given by a weighted sum of *Model Fitness* and *Complexity Fitness*

$$\mathcal{F} = \alpha_1(1 - \mathcal{E}_m) + \alpha_2 \frac{1}{1 + \mathcal{E}_c} \tag{2}$$

with $\alpha_1 + \alpha_2 = 1.0$, and $\mathcal{E}_m$ being the model error. Earlier work [HSM96] showed that α_2 in the range of $0.001 - 0.01$ is sufficient to guide the evolution towards ANNs of low complexity. Thus, we set $\alpha_2 = 0.01$. The model error for classification tasks is simply given by

$$\mathcal{E}_m = \frac{e_v}{n_v}, \tag{3}$$

where e_v is the number of misclassifications on the validation set, and n_v its respective size. TDS fitness is equal to the composite fitness (symbiotic coevolution).

2.2 GA and ANN Parameters

The following GA and ANN parameters have been used with all the experiments in this paper:

GA Parameters: Population Size = 50, Generations = 50, Crossover Probability $p_c = 0.6$, Mutation Probability $p_m = 0.005$, Crossover = 2–Point, Selection Method = Binary Tournament.
ANN Parameters: Network Topology = Generalized Multi–Layer Perceptron, Activation Function (all neurons) = Sigmoid, Output Function (all neurons) = Identity, Training = Error–Back–Propagation, Hidden Neurons (max 10), Learning Parameters η (max 1.0), δ_ϵ (max 0.4), Number of Training Epochs (max 1000), TDS (22 patterns) = Evolutionary, Validation/Test Set = 512 (all possible) patterns.

3 Coevolutionary Scenarios

A simple pattern recognition problem, the detection of lines and their directions, has been adopted from [HMS95]. The input to the ANN is a block of 3×3 black or white pixels, each corresponding to an input neuron. The output layer consists of four neurons activated depending on the line(s) contained in the input pattern. The existence of a known architecture of *minimal* complexity (minANN) (Figure 3) makes this problem amenable for analysis of experimental results.

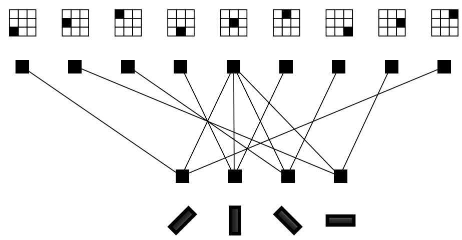

Fig. 3. The lines problem and the corresponding minimal complexity ANN.

For comparisons we define an ANN of *maximal* complexity (maxANN) by a $9-10-4$ fully connected architecture (the possible initial choice of a human designer). In order to train both networks, we use a TDS created by human intuition (provided the authors had some, Figure 4) and an evolutionary generated TDS [MS97].

The minimal and maximal complexity ANN have been trained with each of both TDSs yielding the test set accuracy given in Table 1.

The results are certainly an indicator for the significant interactions between ANN architecture and composition of the TDS. Note that none of the two architectures has been used to generate the evolved TDS, still, it is able to perfectly train the minimal network. It might be even more surprising that the human TDS is not nearly sufficient to train the minimal network, although, its architecture exactly represents the rules defining the four different lines. To further back up this observation, we evolved ANNs for the human TDS (50 generations, 25

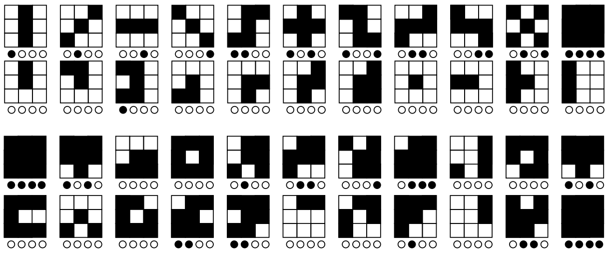

Fig. 4. Human TDS (above) and evolved TDS (below), 22 input patterns and the desired output activations.

	MinANN	MaxANN
Human TDS	0.5527	0.3594
Evolved TDS	1.0000	0.8574

Table 1. Test set accuracy for minimal and maximal ANNs trained by human and evolved TDS (1000 epochs, $\eta = 0.2$, $\delta_\epsilon = 0.1$).

runs), and none of the runs generated a network being able to classify the whole test set correctly (mean accuracy 0.6811 with a standard deviation of 0.0209).

We now look at two extreme scenarios of coevolutionary approaches to the problem at hand. With scenario A we take the maximal network, evolve a TDS for 50 generations, keep the best TDS, and use it as a static environment for the evolution of an ANN (50 generations). Thus, the lifetime of the single environment is 50 times the lifetime of the population of individuals. The scenario is extreme in the sense that it is in fact a sequential evolution of TDSs and ANNs, however, the evolution of the latter is influenced by the former.

The coevolutionary scenario B resembles two independent populations of ANNs and TDSs with equal lifetime evaluating the fitness in a single environment. A TDS is assigned randomly to each ANN, the network is trained, and the fitness of the ANN is equally credited to the TDS. Compared to scenario A this is located on the opposite side of the coevolutionary spectrum, as each individual is ultimately dependent on the single environment it is exhibited to.

In order to keep the number of fitness evaluations (ANN trainings) equal with both scenarios, we ran coevolution for 100 generations with scenario B. Experiments for both scenarios have been repeated 25 times using different random seeds. In all runs an ANN with a test set accuracy of 1.00 could be evolved. A closer look at the coevolved architectures is presented in Table 2.

Generally, both scenarios achieve similar network structures being close to the minimal (optimal) ANN. Scenario B even generates less complex networks (w. r. t. the number of connections) and evolved the best ANN of all runs having only 2 excessive connections compared to the minimal ANN. Typically, the excessive connections are trained to very small weights ($0.01 - 0.1$), thus,

	Hidden	Connections
Mean	0.80 / 1.20	29.72 / 25.36
Min	0 / 0	17 / 14
Max	2 / 4	45 / 45
StdDev	0.7071 / 1.1902	6.2418 / 8.1949

Table 2. Statistics of coevolved ANN architectures of scenarios A / B (averaged on 25 runs).

ANN training additionally acts as a local (structural) optimization procedure. The random placement of individuals in a single environment (scenario B) is a rather brute method, as "good" ANNs might not survive a "bad" TDS (or vice versa). However, it also ensures the evolution of robust structures being able to deal with a hostile environment. This is indicated by the pattern frequency in the concatenated best TDSs evolved in scenario A and scenario B (Figure 5).

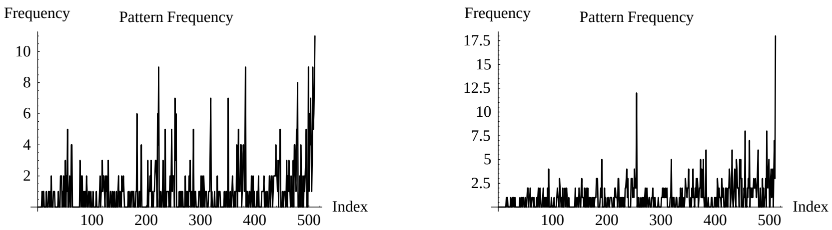

Fig. 5. Pattern frequency of concatenated best TDSs (25 runs) in scenario A (left) and scenario B (right).

While the patterns contributing to the best TDSs found in scenario A (single ANN, static environment) are widely spread, scenario B produces a more pronounced selection of (robust) patterns. Specifically, pattern 511 [4] (all black pixels), and pattern 255 (all black but upper left corner) contribute positively to TDSs for a great variety of architectures. Both patterns can also be found in the previously discussed evolved TDS (Figure 4).

Summarizing, we may say that symbiotic coevolution of ANNs and TDSs can improve the performance of ANNs utilized for classification tasks. This could be specifically useful for prediction problems (e.g., stock markets), where the available data change (increase) with time and a neural predictor could be coevolved (dynamically adapted) to new data.

[4] The index is the decimal representation of the binary input vector.

4 Outlook

The differences between the two extremes of coevolutionary scenarios studied in this paper are rather small suggesting similar results for scenarios in between. Nevertheless, the variation of lifetime and number of environments has to be investigated in combination with more difficult (real world) problems in the future. Among current research topics is the extraction of rules from ANNs generated by netGEN, as the low complexity architectures might enable the generation of small, consistent rule bases extracting the knowledge accumulated by evolution.

5 Acknowledgements

We wish to thank the *Austrian Center for Parallel Computation* (ACPC), Group Salzburg, for making available a cluster of DEC AXP workstations for this research. Also, we would like to thank our students Karl Fürlinger and Marc Strapetz for valuable support in data acquisition and presentation.

References

[Bar97] Helio J. C. Barbosa. A coevolutionary genetic algorithm for a game approach to structural optimization. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 545–552, San Francisco, California, 1997. Morgan Kaufmann Publishers, Inc.

[GBD+94] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam. *PVM 3 User's Guide and Reference Manual.* Oak Ridge National Laboratory, 1994.

[Hil90] W. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42:228–234, 1990.

[HMS95] Reinhold Huber, Helmut A. Mayer, and Roland Schwaiger. netGEN - A Parallel System Generating Problem-Adapted Topologies of Artificial Neural Networks by means of Genetic Algorithms. In *Beiträge zum 7. Fachgruppentreffen Maschinelles Lernen der GI-Fachgruppe 1.1.3, Forschungsbericht Nr. 580, Dortmund*, August 1995.

[HP89] S.J. Hanson and L.Y. Pratt. Comparing biases for minimal network construction with back-propagation. In D. Touretzky, editor, *Advances in Neural Information Processing Systems, Vol. 1*, volume 1, pages 177–185. Morgan Kaufmann, 1989.

[HSM96] Reinhold Huber, Roland Schwaiger, and Helmut A. Mayer. On the Role of Regularization Parameters in Fitness Functions for Evolutionary Designed Artificial Neural Networks. In *World Congress on Neural Networks*, pages 1063–1066. International Neural Network Society, Lawrence Erlbaum Associates, Inc., 1996.

[Lev91] James R. Levenick. Inserting Introns Improves Genetic Algorithm Sucess Rate: Taking a Cue from Biology. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 123–127. University of California, San Diego, Morgan Kaufmann, 1991.

[Lov88] James Lovelock. *The Ages of Gaia: A Biography of Our Living Earth.* W. W. Norton, 1988.

[May97] Helmut A. Mayer. *ptGAs - Genetic Algorithms Using Promoter/Terminator Sequences - Evolution of Number, Size, and Location of Parameters and Parts of the Representation.* PhD thesis, University of Salzburg, 1997.

[MS97] Helmut A. Mayer and Roland Schwaiger. Towards the Evolution of Training Data Sets for Artificial Neural Networks. In *Proceedings of the 4th IEEE International Conference on Evolutionary Computation*, pages 663–666. IEEE Press, 1997.

[MTH89] Geoffrey F. Miller, Peter M. Todd, and Shailesh U. Hegde. Designing Neural Networks using Genetic Algorithms. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 379–384, San Mateo, California, 1989. Philips Laboratories, Morgan Kaufmann Publishers, Inc.

[Par94] Jan Paredis. Steps towards Co–evolutionary Classification Neural Networks. In R. Brooks and P. Maes, editors, *Proceedings Artifical Life IV*, pages 545–552. MIT Press / Bradford Books, 1994.

[PJ95] Mitchell A. Potter and Kenneth A. De Jong. Evolving Neural Networks with Collaborative Species. In *Proceedings of the 1995 Summer Computer Simulation Conference*, 1995.

[Plu94] Mark Plutowski. *Selecting Training Examplars for Neural Network Learning.* PhD thesis, University of California, San Diego, 1994.

[RG93] Nicholas J. Radcliffe and Felicity A. W. George. A Study in Set Recombination. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 23–30. University of Illinois at Urbana-Champaign, Morgan Kaufmann, 1993.

[Röb94] A. Röbel. The Dynamic Pattern Selection: Effective Training and Controlled Generalization of Backpropagation Neural Networks. Technical report, Technical University Berlin, 1994.

[SGE91] Robert E. Smith, David E. Goldberg, and Jeff A. Earickson. SGA-C: A C–Language Implementation of a Simple Genetic Algorithm. TCGA Report 91002, The Clearinghouse for Genetic Algorithms, The University of Alabama, Department of Engineering Mechanics, Tuscaloosa, AL 35487, May 1991.

[Wu96] Annie Siahung Wu. *Non–Coding DNA and Floating Building Blocks for the Genetic Algortihm.* PhD thesis, University of Michigan, 1996.

[Zha93] Byoung-Tak Zhang. Self–development learning: Constructing optimal size neural networks via incremental data selection. Tech. Rep. No. 768, German National Research Center for Computer Science, Sankt Augustin, 1993.

[Zha94] Byoung-Tak Zhang. Accelerated Learning by Active Example Selection. *International Journal of Neural Systems*, 5(1):67–75, 1994.

[ZMV+94] Andreas Zell, Guenter Mamier, Michael Vogt, Niels Mach, Ralf Huebner, Kai-Uwe Herrmann, Tobias Soyez, Michael Schmalzl, Tilman Sommer, Artemis Hatzigeogiou, Sven Doering, and Dietmar Posselt. *SNNS Stuttgart Neural Network Simulator, User Manual.* University of Stuttgart, 1994.

This article was processed using the LaTeX macro package with LLNCS style