Evolutionary Structuring of Artificial Neural Networks

H.–M. Voigt, J. Born, I. Santibáñez–Koref

Technical University Berlin
Bionics and Evolution Techniques Laboratory
Bio– and Neuroinformatics Research Group
Ackerstraße 71–76 (ACK1)
D–1000 Berlin 65

e–mail: bbc@max.fb10.tu-berlin.de

Abstract

The report summarizes research on the structuring of Artificial Neural Networks by a stochastic graph generation grammar. The main feature of our approach is to carry out the graph generation in view of an individual development process which is embedded in an evolutionary process. We intend to explain our approach by examples, and to evaluate its practicability.

1 Introduction

The general purpose of our approach – in a mathematical sense – is to optimize pseudo–Boolean functions (i.e. real–valued functions, depending from binary variables). In this report, we regard only one problem which belongs to this class of optimization problems: the structuring of Artificial Neural Networks (ANN). But, at the end of the report we shall return to the general purpose in context with conclusions on the practicability of our method.

Among the natural models used to process information, that of ANN has become one of the most used in the last years. Only a few remarks about ANN will be made, and further references can be found in literature, e.g. [13].

One of the purposes of ANN is function approximation: Let be $\Psi : \Re^{N_I} \rightarrow \Re^{N_O}$. The following system of (unit) functions characterizes a Feed–Forward Net (FFN):

- $U_i(I) = I_i$ for $i = 1, ..., N_I$ (input–units)
- $U_i(I) = f(\sum_{j=1}^{i-1}(w_{ij}U_j(I) + h_i)$ for $i = N_{N_I+1}, ..., N_D$ where is $f : \Re \rightarrow \Re$ and $w_{ij}, h_i \in \Re$ (hidden-units, w_{ij} are called weights or connections between the units)
- $O_i(I) = U_{N_D - N_O + i}$ for $i = 1, ..., N_O$ (output–units)

The only information which we have about Ψ is a set of samples $S = \{(I^k, \Psi(I^k)),$ $k = 1, ..., N_S\} \subset \Re^{N_I} \times \Re^{N_O}$. The task is to approximate Ψ with the FFN using the sample S. This is done by adapting the weights w_{ij} and h_i in such a way that a certain error criteria (e.g. the mean square error) between the output $O(I^k) = (O_1(I^k), ..., O_{N_O}(I^k))$ of the FFN and the corresponding sample $\Psi(I^k)$ is minimized. This procedure is called training of the FFN.

The matrix:

$$c_{ij} = \begin{cases} 1 & : \quad w_{ij} \neq 0 \quad \text{and} \quad i < j \\ 0 & : \quad else \end{cases} \qquad \text{with} \quad i, j = 1, ..., N_D$$

is the connectivity matrix of the FFN. The matrix c can be interpreted as an adjacency matrix of a directed graph which represents the network structure.

Setting all w_{ij} identical to zero for which the c_{ij} is zero, the training procedure will try to find optimal values for all other weights.

In the following, we describe a model of individual development based on Lindenmayer–Systems (L–Systems) [9], the Building Blocks in Cascades Algorithm (BBC–Algorithm).

Section 2 gives a detailed description of our approach. In section 3, we describe briefly the evolutionary framework (the BBC–EVO–Algorithm), a version of the BBC-Algorithm embedded in an evolutionary process. Section 4 presents the BBC–EVO–Algorithm in comparison to the former BBC–Algorithm with fixed parameters, tested on a target–matrix. Section 5 shows some exemplary results in order to describe the performance of the BBC-EVO-Algorithm. In section 6, we explain the mechanics of the BBC–Algorithm by a hypothetical example. Finally, in section 7 we draw some conclusions from our current work.

2 Recursive Generation of Connection Structures – the BBC–Algorithm

Kitano [7] has introduced grammatical encoding of ANN. He assumes that the connectivity matrix of a FFN can be described by applying a deterministic context–free Graph–L–System to an axiom. With a genetic algorithm he generates and selects such productions which lead to a graph generation grammar with good problem solving properties. This grammar generates within m recursive steps a $(2 \times 2)^m$ connectivity matrix by rewriting words by words in form of (2×2) matrices with an initial (1×1) matrix (axiom).

Mjolness [10] derives a problem–independent recursion equation for connectivity matrices.

Gruau [4], [5] was inspired by Kitano's approach. He rewrites "cells" of networks, instead of rewriting words. A Genetic Algorithm (applying Koza's Genetic Programming paradigm [8]) is used to recombine grammar trees representing cellular encodings.

Our approach (basics are presented in [14]) was also inspired by Kitano's paper. We use a stochastic L–System. The productions are probability dependent. This is a decisive difference to Kitano's approach. The application of stochastic L–Systems avoids the generation of high redundant sets of productions.

The basic algorithm is a recursive schema as used by Kitano:

After an initial generation of a directed tree by recursive $(2 \times 2)^i$–steps, i= 1,...,m, this tree is iteratively and randomly modified by changing sub–trees.

Thus, the generation of the final matrix can be imagined as a sequence of 2×2 cascades.

Therefore, we call the algorithm the "Building Blocks in Cascades" algorithm (BBC–Algorithm).

In our paper [14], we proposed a cyclic scheme with fixed control parameters to apply the BBC–Algorithm. The following algorithmic version allows the adaptation of the control parameters – e.g. by an evolutionary process. In the following, we will denote this former version, the Cyclic BBC– Algorithm (C–BBC– Algorithm).

2.1 Algorithmic Description of the BBC–Algorithm

Let $G = (K, B)$ be a rooted tree.

$K \subseteq \Sigma, \Sigma \neq \emptyset$ is a set of vertices and $B \subset K \times K$ a set of directed edges between vertices from K.

Let $SUC(k_0) = \{k \mid \exists b_1, ..., b_n \in B, b_1 = (k_0, k_1), b_2 = (k_1, k_2), ..., b_n = (k_{n-1}, k)\}$ be the set of successors of k_0.

All vertices with a distance from the root equal to i are called vertices of the i^{th} level.

Let $V = (\Sigma, p, w, m)$ with the axiom $w \in \Sigma$ be a tree generation process , i.e. the root of the tree to be generated. Let be $p : \Sigma \times \Sigma \times \Sigma \rightarrow \Sigma \times \Sigma \times \Sigma$ a generation instruction and m the recursion depth of the generation process.

The generation instructions are defined as follows:

$$G_0 = (K_0, B_0)$$

with $K_0 = \{w\}$, $B_0 = \emptyset$

$$G_1 = p(G_0) = (K_1, B_1)$$

with $K_1 = K_0 \cup \{k_{1,1}, k_{1,2}, k_{1,3}, k_{1,4}\}$, $B_1 = B_0 \cup \{(w, k_{1,1}), (w, k_{1,2}), (w, k_{1,3}), (w, k_{1,4})\}$

...

$$G_i = p(G_{i-1}) = (K_i, B_i)$$

with $K_i = K_{i-1} \cup \{k_{i,1}, ..., k_{i,4^i}\}$, $B_i = B_{i-1} \cup \{(k_{i-1,j}, k_{i,4(j-1)+k}) \quad j = 1, ..., 4^{i-1}, k = 1, ..., 4\}$

...

$$G_m = p(G_{m-1}) = (K_m, B_m)$$

with $K_m = K_{m-1} \cup \{k_{m,1}, ..., k_{m,4^m}\}$, $B_m = B_{m-1} \cup \{(k_{m-1,j}, k_{m,4(j-1)+k}) \quad j = 1, ..., 4^{m-1}, k = 1, ..., 4\}$

Let $A = \{AT : K \rightarrow \{0, 1\}\}$ the set of mappings which assigns to every node an element from the set $\{0, 1\}$ (coloring of the tree) where $\{0, 1\}$ is called the set of attributes of the nodes.

Then the following functions are defined:

- Let be $flip : \{0, 1\} \rightarrow \{1, 0\}$ with $flip(0) = 1$ and $flip(1) = 0$
- Let $FLIP : K \times A \rightarrow A$ with

  $$FLIP(k_0, AT)(k) = \begin{cases} flip(AT(k_0)) & k = k_0 \\ flip(AT(k_0)) & k \in SUC(k_0) \\ AT(k) & else \end{cases}$$

  be a function which flips the attribute of a node and rewrites the attributes of the successor nodes with this attribute.
- Let $COPY : K \times A \rightarrow A$ with

  $$COPY(k_0, AT)(k) = \begin{cases} AT(k_0) & k \in SUC(k_0) \\ AT(k) & else \end{cases}$$

  be a function which copies the attribute of a node to all successor nodes.

Then the BBC–Algorithm can be described as follows:

- *Initial step:*

  Set the recursion depth m.

  Generate a tree with the tree generation process V.

  Choose a coloring AT of the tree.

  Determine for every level l_i probabilities p_i (probability for the application of the FLIP function) and f_i (probability for the application of the COPY function).

- *BBC step:*

  Choose with probability l_i a level i and perform for the nodes of the i^th level:

  - Choose with probability p_i a node k and set $AT' = FLIP(k, AT)$.
  - Choose with probability f_i a node k and set $AT'' = COPY(k, AT')$.
  - Set the final connectivity matrix by the attributes of the terminal nodes K_m corresponding to AT''.

3 Evolutionary Framework – The BBC–EVO–Algorithm

We embedded the BBC–Algorithm in an evolutionary framework. This was based on the following problem. As for all random search methods, the efficiency of the BBC–Algorithm depends on a well–setting of the control parameters. A good choice of these parameters depends on the (unknown) topology of the problem which is to be solved.

An evolutionary framework for the BBC–Algorithm permits a dynamical adaptation of the control parameters.

The evolutionary framework signifies: Start with an initial population. Each individual of this population is described by the control parameters of the BBC–Algorithm. The control parameters (the BBC–Parameters) are l_i, p_i, and f_i, $i = 1, ..., m$, used in the BBC–Step.

A new population is generated by recombination and mutation of individuals of the parent population. Each member of the new population is evaluated in order to assign a fitness to it. Based on this evaluation a selection procedure takes the worst individuals from the following evolutionary process. Thus, only the individual with the best fitness will survive. By iterating this procedure a progress in the sense of finding better solutions is obtained.

The general framework is sketched in Figure 1.

Each individual undergoes a development, by applying the BBC–Algorithm controlled by the BCC–Parameters. The BBC–Parameters characterizes the genotype of an individual. This leads to a the so–called **preformed phenotype** and is built by the resulting connectivity matrix, and the variables of its recursive description. The **adult phenotype** is the reduced connectivity matrix, generated by deletion of unused connections (We consider only the upper right triangle for feed–forward networks. Unused connections are connections that are not located on a path between input and output in the corresponding network–graph.). This resulting matrix is transformed into a FNN, which is trained with a given sample set S, to see how well the FFN performs the required task.

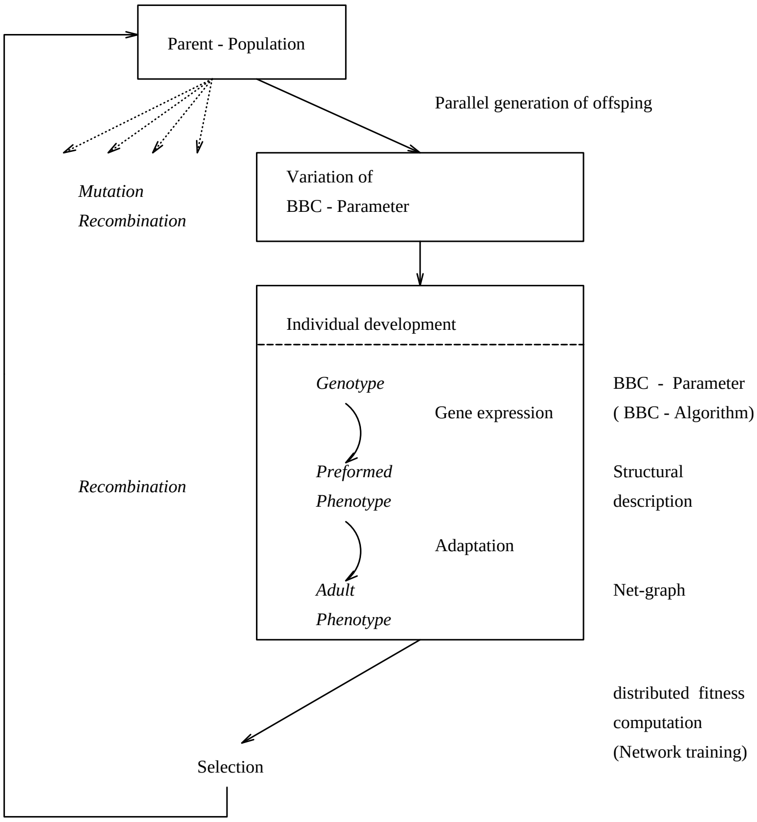

Figure 1: Evolutionary and Developmental Scheme

The genetic operators are:

- Mutation and recombination on genotype level.
- Recombination on level of the preformed phenotypes.
- Selection is determined by the fitness of the adult phenotype.

The main feature is the individual developmental process and is characterized by two stages: First, the BBC–Algorithm representing the individual growth and second, the training of the resulting FFN resulting from the adult phenotype in order to determine the fitness of the individual.

The evolutionary framework is based on the classical $(\mu \overset{+}{,} \lambda)$ Evolution Strategy (μ – number of parents, λ – number of offspring; see e.g. [11], [12]), with self–adaptation of the strategy parameters. The classical scheme is parallelized on generation level for distributed fitness evaluation.

We denote this procedure the BBC-EVO – Algorithm.

4 The BBC–EVO–Algorithm in Comparison to the BBC–Algorithm with Fixed Parameters

To check the consistency of the BBC–EVO–Algorithm with the former C–BBC– Algorithm, the approximation of a target matrix is analyzed . As example we use a (64×64) target matrix, as shown in Figure 2.

To evaluate the approximation performance we proceed as we have it done in [14].

The fitness is computed as the weighted sum of differences of the elements of the upper triangular matrix. For 1–blocks high weights are used. Elements with increasing distance from 1–blocks are also increasingly weighted.

The 1–blocks, which have to be determined, have the dimensions (16×16), (8×16), (8×20), (20×4).

This connectivity matrix represents a FFN with three hidden layers where the first layer has four receptive fields.

For the determination of this structure operations up to the 5^{th} level of the generating tree have to be applied, i.e. (4×4)–units must be manipulated.

The C–BBC–Algorithm is applied with the default control parameters as used in [14] .

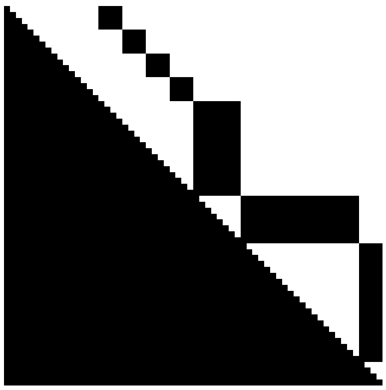

Figure 2: Target Matrix ; (In this and the next figures of connectivity matrices: no connection (0) in the upper right triangle is white, connection (1) is black. (The lower left triangle is uninteresting, and black.))

The results for finding the exact solution are listed in the following Table. (The numbers of generated individuals are presented.) The notation $(\mu \overset{+}{,} \lambda)$– BBC–EVO corresponds with the scheme of a $(\mu \overset{+}{,} \lambda)$ Evolution Strategy.

	best	worst	average
C–BBC	9,138	47,621	21,774
(1+1)–BBC–EVO	6,941	45,733	21,560
(30+200)–BBC–EVO	1,400	25,800	21,380
(30,200)–BBC–EVO	21,600	50,000	32,020

We emphasize that in this case of approximation a target matrix, there is no difference between the connectivity matrices of the preformed and the adult phenotype.

The results should not be misinterpreted. The default setting of the control parameters for the C–BBC– Algorithm is adequate in the case of approximation of target matrices. We see the results for (1+1)– and (30+200)– BBC–EVO are better than for the C–BBC. The simulation runs have shown that for the (1 + 1)– and (30 + 200)– BBC–EVO the BBC–Parameters were adapted rapidly for operating on deep levels l_i and with small probabilities p_i, f_i. This is optimal for this task.

We shall only mention here that the BBC–Parameters are mutated in the BBC–EVO–Algorithm normally distributed within fixed boundaries, and using fixed "strategy variances". The boundaries and the strategy variances are chosen in the initial step of the BBC–EVO and will be held constant. By this procedure, the problem to choose well–set control parameters is transferred to a higher level. But we will not discuss this here, and refer to [11], [12]).

(30,200)–BBC–EVO has a worse performance. The reason is that we use large strategy variances, and in this version the selection pressure is too low (the BBC– Parameters were not stabilized for deep levels l_i and small probabilities p_i, f_i).

5 The BBC–EVO–Algorithm as a Universal Network Generator

In order to gain experiences, we performed experiments with the approximation of target–matrices. For results see [14], and the above mentioned tests. In this case the used fitness measure of an individual is a modified hamming-distance between the target–matrix and the connectivity matrix of the adult phenotype.

In order to generate ANN, we have to choose a new fitness criterion. The fitness should be determined by mixing the following measures of the adult phenotype.

- (Mean square) learning–error
- Classification–error
- Number of iterations for the training of the network
- Number of connections in the network–graph
- Minimal path–length in the network–graph [1]
- Maximal path–length.

5.1 Exemplary Tests

The following tests with the well known parity problem (see [13]) of dimension 3 should demonstrate the characteristic features of the BBC–Algorithm. For the tests, we had used 10 workstations for the distributed fitness evaluations (by supervised learning of the generated networks using a back–propagation procedure, see [13]).

The run–times were 8,819 seconds for the first example and 312,201 seconds for the second example.

The tests were performed by a $(20 + 100)$–version of the BBC–EVO, and we used 10 resp. 8 trials for each problem.

The recursion depth of the BBC–Algorithm was 5, i.e. the connectivity matrix had the dimension (32×32).

For the back–propagation procedure, a maximum 10,000 iterations were set.

[1]Path–length is the number of vertices between input and output within the network–graph, so that no node occurs twice.

5.1.1 Problem 1: 3-Dimensional Parity Problem with Minimization of the Number of Connections in the Network–Graph

The fitness function for the adult phenotype was computed as the weighted sum of the learning–error and the number of connections in the network–graph. [2]

A trial was stopped if the best individual of the population was represented by an adult phenotype with a correctly trained network, having 10 or less connections. The results are listed in the following Tables.[3]

problem 1	best	worst	average
Number of connections	7	10	–
Minimal path–length	1	1	–
Maximal path–length	2	3	–
Number of generations	2	7	4

problem 1	total
Number of lethal adult phenotypes	278
Number of vital phenotypes with a correctly trained network	3,730
Number of vital phenotypes with an unsuccessfully trained network	192

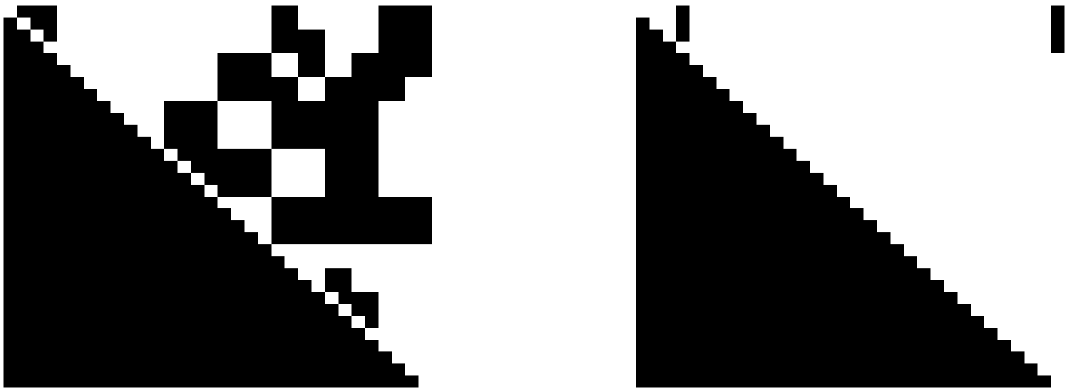

Figure 3: The best unreduced and reduced connectivity matrix for problem 1.

Figure 3 shows the best connectivity matrix and the corresponding reduced connectivity matrix. Here, and for the following figures of reduced connectivity matrices, a "normed" or "renumbered" reduced connectivity matrix is shown. This reduced connectivity matrix corresponds with the reduced connectivity matrix generated by deletion of unused connections. Only the nodes and connections are newly numbered from the beginning.

Figure 4 shows the network graph corresponding to the best connectivity matrices for problem 1.

[2]The weight for the learning–error was 0.0 for correctly trained networks, else 1000.0; the weight for the number of connections was 1.0.

[3]We denote an adult phenotype as lethal if there is no path between input and output within the corresponding network

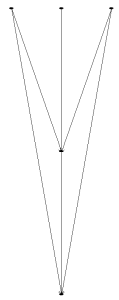

Figure 4: Network with 1 hidden unit within 1 hidden layer, and 7 connections

5.1.2 Problem 2: 3–Dimensional Parity Problem with Generation of a Deep–Layered Network with Few Connections in the Network–Graph

The fitness function for the adult phenotype was computed as the weighted sum of the learning–error, the (negative) minimal path–length, the (negative) maximal path–length, and the number of connections in the network–graph. [4]

A trial were stopped after 100 generations. The results are listed in the following Table:

problem 2	best	worst	average
Number of connections	105	102	-
Minimal path–length	9	7	-
Maximal path–length	29	25	-
Number of generations	100	100	-

problem 2	total
Number of lethal adult phenotypes	14,850
Number of vital phenotypes with a correctly trained network	62,315
Number of vital phenotypes with an unsuccessfully trained network	2,995

[4] The weight for the learning–error was 0.0 for correctly trained networks, otherwise 1000.0; the weight for the minimal path–length was -10.0; the weight for the maximal path–length was -0.5; the weight for the number of connections was 0.005.

Figure 5 show the best connectivity matrix and the corresponding reduced connectivity matrix for problem 2.

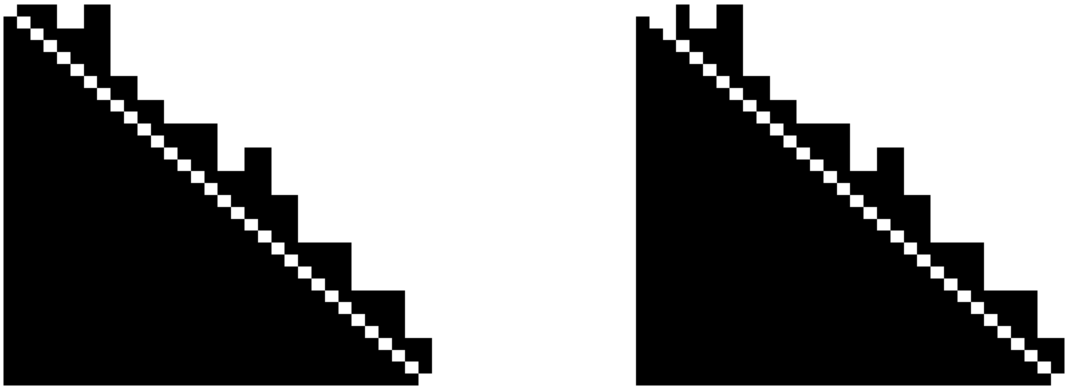

Figure 5: The best unreduced and reduced connectivity matrix for problem 2

Figure 6 shows the network graph corresponding to the best connectivity matrices for problem 2.

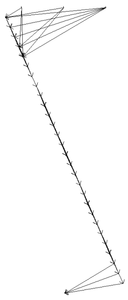

Figure 6: Network with 28 hidden units within 28 hidden layers, and 105 connections. (The connections between the hidden nodes are merged.)

5.1.3 Evaluations

Both test problems indicate of best–case and worst–case of efficiency in generating desired topologies of networks.

The main basis of the different rates of convergence, demonstrated by the results for problems 1 and 2, is the different probability of success, i.e. the chance to generate an offspring, having an adult phenotype with a better fitness than the fitness of the parents.

The probability of success is high for the first problem at the beginning and during the evolutionary progress. Figure 3 shows that in this case the BBC–Algorithm has to operate only with a recursion depth of 3; i.e. a **minimal block–size** of (4×4) is sufficient to generate the best adult phenotype. Furthermore, figure 3 illustrates that on the level of the preformed phenotype there is a high degree of **redundancy** to generate the adult phenotype. Redundancy within the connectivity matrix of the preformed phenotype, concerning the corresponding reduced connectivity matrix of the adult phenotype, means that there are "blocks", which can vary without an effect on the fitness. Due to this, the same connectivity matrix of the adult phenotype can be generated by different connectivity matrices of the preformed phenotype.

Both factors - the minimal block size and the redundancy - generate a high probability of success and therefore a high rate of convergence.

For problem 2 there exists after some few generations a low probability of success, therefore a small rate of convergence. As soon as in the connectivity matrix of the preformed phenotype all nodes are connected (the maximization of the path-length is done), all 1–blocks will also emerge in the adult phenotype, and will deteriorate the fitness.

Figure 5 may illustrate that at the end of the evolutionary process, a success would be only possible by operating on the 5^{th} level. The minimal block size decreases to (1×1), and the redundancy disappears. A low convergence rate follows.

5.2 Tests with Higher Dimensions of Problem 1

The following tests with parity problems of sizes 5, and 7 should demonstrate the relatively good "scaling property" – with regard to the number of generations needed to generate a sub–minimal network. We do not present a functional relation between the problem size and the number of generations. The amount of generations depends on the "matching" of an optimal unreduced connectivity matrix to the recursion scheme, and not directly on the problem size.

5-Dimensional Parity Problem

The fitness function for the adult phenotype was computed as the weighted sum of the learning–error and the number of connections in the network–graph. A trial was stopped if the best individual of the population was represented by an adult phenotype with a correctly trained network, having 20 or less connections. For the back–propagation procedure, as a maximum 30,000 iterations were set.

The results are listed in the following Table:

par5	best	worst	average
Number of connections	18	20	–
Minimal path–length	1	1	–
Maximal path–length	3	4	–
Number of generations	4	13	7

Figure 7: The best unreduced and reduced connectivity matrix for problem par5

Figure 7 show the best connectivity matrix and the corresponding reduced connectivity matrix for problem par5. Figure 8 shows the network graph corresponding to the best connectivity matrices for problem par5.

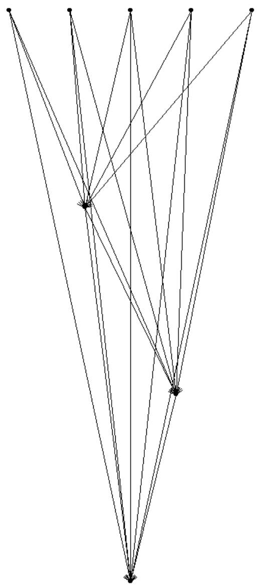

Figure 8: Network with 2 hidden units within 2 hidden layers, and 18 connections

7-Dimensional Parity Problem

The fitness function for the adult phenotype was computed as the weighted sum of the learning–error and the number of connections in the network–graph.

A trial was stopped if the best individual of the population was represented by an adult phenotype with a correctly trained network, having 34 or less connections or after 10 generations.

For the back–propagation procedure, as a maximum 30,000 iterations were set.

The results are listed in the following Table:

par7	best	worst	average
Number of connections	34	39	-
Minimal path–length	1	1	-
Maximal path–length	4	2	-
Number of generations	1	10	-

Figure 9 show the best connectivity matrix and the corresponding reduced connectivity matrix for problem par7. Figure 10 shows the network graph corresponding to the best connectivity matrices for problem par7.

Figure 9: The best unreduced and reduced connectivity matrix for problem par7

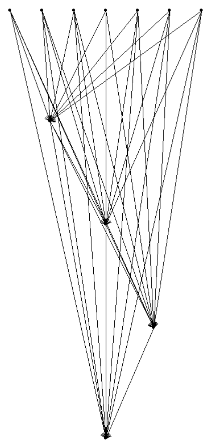

Figure 10: Network with 3 hidden units within 3 hidden layers, and 34 connections

5.3 Tests for the 2–D, Two Spirals Problem

The 2–dimensional, two spirals problem was posted to the "connectionist" mailing list by A. Wieland of MITRE Corporation.

Figure 11 illustrates this problem. The task is to learn to discriminate between two sets of training points which lie on two distinct spirals in the x–y plane. These spirals coil three times around the origin and around one another.

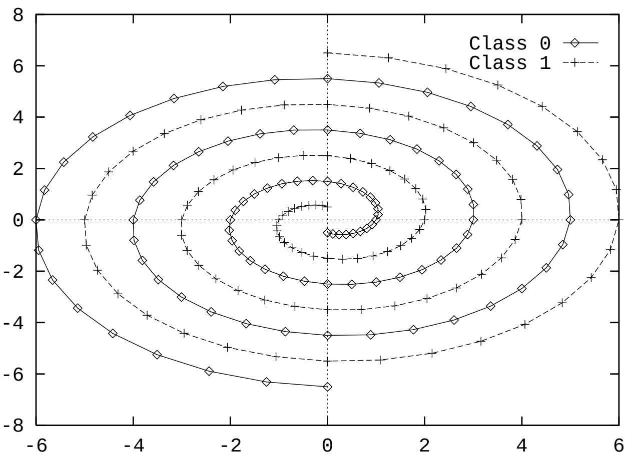

Figure 11: 2–dimensional, two spirals problem

This appears to be a very difficult task for back–propagation networks.

Fahlman has summarized some results obtained on this problem in [2]. The results, cited in his report, can be found in [6], [1], and [3].

Among the cited results, the most elegant and easily found solution is generated by the Cascade–Corellation algorithm by Fahlman and Lebiere [3]. This algorithm is able to solve this problem in 1,700 epochs (average over 100 trials). In these trials, the networks constructed ranged from 12 to 19 hidden units, with a median of 15.

We used a (6+30)–BBC–EVO scheme to solve the problem. The recursion depth of the BBC–Algorithm was 5, i.e. the connectivity matrix had the dimension (32×32). For the back–propagation procedure, a maximum 10,000 iterations were set.

The fitness function for the adult phenotype was computed as the weighted sum of the pattern–error and the number of connections in the network–graph. [5] We set as lower bound for correctly trained networks a pattern–error of 5% .

We used 10 trials, and a trial were stopped after 10 generations. The results are listed in the following Tables:

Two spiral problem	best	worst
Number of connections	44	96
Minimal path–length	1	1
Maximal path–length	8	11
Number of BP–iterations	1,150	4,250

Two spiral problem	total in the best run	total in the worst run
Number of lethal adult phenotypes	69	42
Number of vital phenotypes with a correctly trained network	156	196
Number of vital phenotypes with an unsuccessfully trained network	81	68

The second Table shows a high rate of vital phenotypes which could not be trained. This result was arised from the generation of shallow networks with a small number of connections. The simulation results show that such a kind of networks are hard to train.

The results confirm that deep layered networks are optimal for this problem.

The BBC–EVO rapidly generates this type of networks.

[5] The weight for the pattern–error was 0.0 for correctly trained networks, else 1000.0; the weight for the number of connections was 1.0.

Figure 12: The best unreduced and reduced connectivity matrix for the two spirals problem

Figure 12 shows the best connectivity matrix and the corresponding reduced connectivity matrix for the two spiral problem. Figure 13 shows the network graph corresponding to the best connectivity matrices for the two spiral problem.

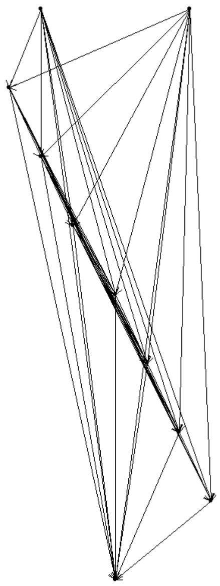

Figure 13: Network with 7 hidden units within 7 hidden layers, and 44 connections. (The connections between the hidden nodes are merged.)

6 Analysis of the Mechanics

In this chapter, we will try to illustrate why our algorithm becomes so effective for problem 1. We will explain the mechanics by an hypothetical example.

Figure 14 illustrates the best reduced connectivity matrix found for problem 1 within the 3^{rd} or $((4 \times 4)$ –) level of the recursion scheme. We mark 1–blocks (connections are present) dark, else 0–blocks (no connections) are illustrate. The lower left triangle is only shown if parts of it are relevant.

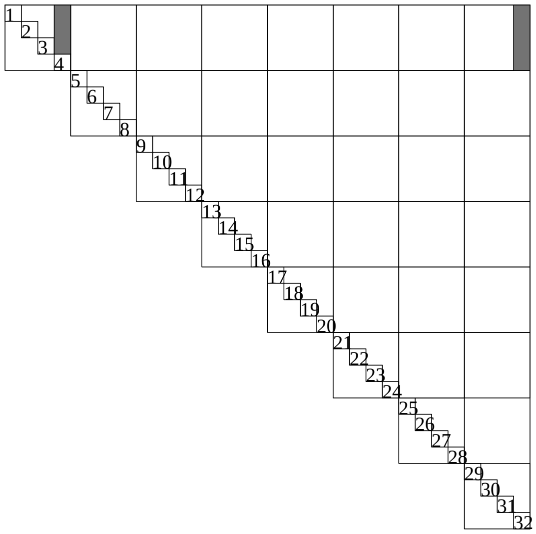

Figure 14: Best reduced connectivity matrix for problem 1 within the (4×4) – level

A hypothetical unreduced connectivity matrix which generates this best structure is shown in Figure 15. Many unreduced matrices form this best structure, among these is the evolved unreduced connectivity matrix, shown in Figure 3. In the figure, the 1–blocks of the unreduced matrix are dark, the 1–blocks of the resulting reduced matrix are darker. The reduced structure is formed in the illustrated manner, first, no node of the nodes 17 – 31 have a connection to the output node 32, and, second, the (4×4)–1–block in the upper left corner is reduced by the connections between the three input nodes.

We construct a way to generate this hypothetical unreduced connectivity matrix in three steps.

The BBC–EVO–Algorithm may start with the coloring "1", i.e. we start with connectivity matrices, completely filled by 1–blocks.

We regard the following possible variations within the (best) unreduced connectivity matrix.

1^{st} variation: operation on the 1^{st} level, flip the (16×16) –block **A** to a 0–block – Figure 16.

2^{nd} variation: operation on the 2^{nd} level, flip the (8×8) -block **B** to a 0–block – Figure 17.

3^{rd} variation: operation on the 3^{rd} level, flip the (4×4) -block **C** to a 0–block – Figure 18.

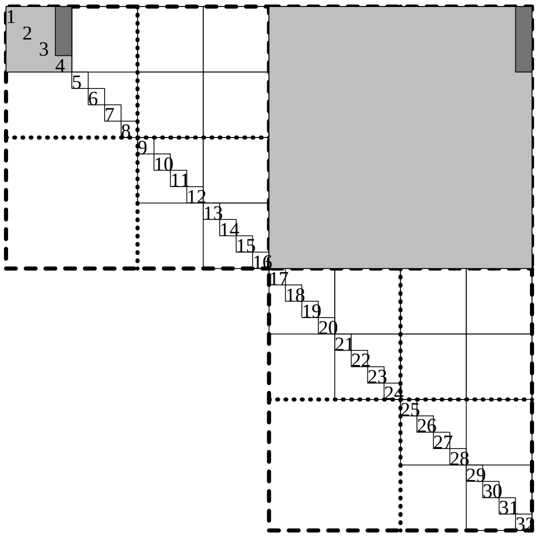

Figure 15: A hypothetical unreduced connectivity matrix forming the founded best reduced connectivity matrix for problem 1

The three variations are stochastically independent, they can be carried out in any sequence whatever, and could improve the fitness.

The probabilities to carry out the variations are: for the 1^{st} variation $= 1/4 \times 1/5 = 1/20$, for the 2^{nd} variation $= 1/16 \times 1/5 = 1/80$, and for the 3^{rd} variation $= 1/64 \times 1/5 = 1/320$.

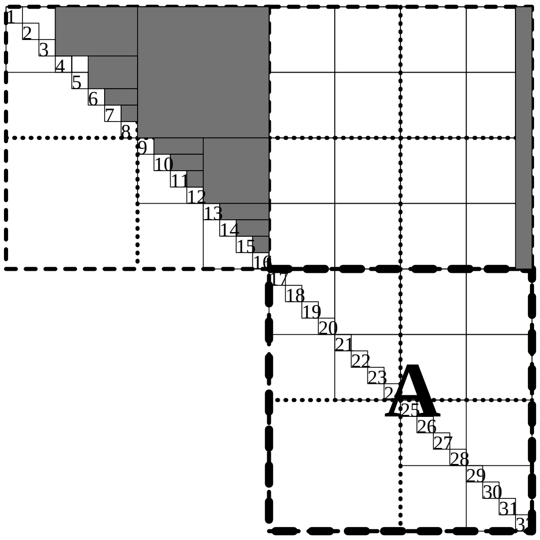

Figure 16: Reduced connectivity matrix after variation of block A

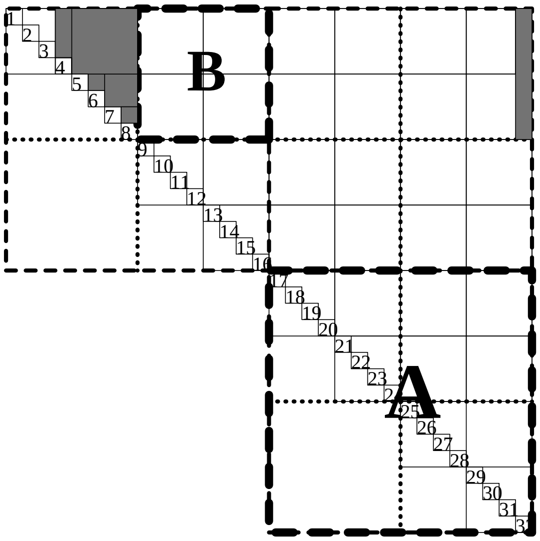

Figure 17: Reduced connectivity matrix after variation of blocks A, and B

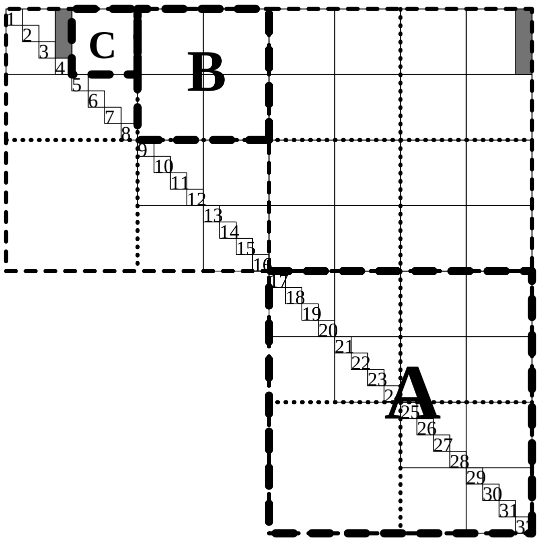

Figure 18: Reduced connectivity matrix after variation of blocks A, B, and C

Even the probability of occurrence of this sequence of only three variations is low. But, the hypothetical example is only one possible way leading to the optimum. The redundancy within the unreduced connectivity matrices allows to generate many paths to the optimum. In the average, the BBC–EVO has needed only 420 fitness evaluations!

7 Conclusions

We have seen that the BBC–EVO–Algorithm can be used as a universal network generator.

The recursive generation scheme does not use of any context information from the training procedure. The solving behavior of the special problem has only an implicit feedback to BBC–Algorithm. The BBC–Parameters will adapted to successful variations.

The method will be efficient if the "hidden structure" of the problem allows the features of the BBC–Algorithm to be effective. The exemplary test problems illustrate this.

Generally, the BBC–Algorithm is a random search procedure to optimize pseudo–Boolean functions.

The BBC–EVO– Algorithm can be regarded as a subspaces–oriented random search method, using an external structuring of the search space by the $(2 \times 2)^i$ – recursion scheme.

It offers an efficient search method in case of:

- Block–structured problems (i.e. problems, where (optimal) variables have the same values in segments (blocks)).

and/or

- Redundant problems (i.e. problems, where for (optimal) variables the first derivatives are null within subspaces).

8 Acknowledgement

This research was supported by the Bundesministerium für Forschung und Technologie (BMFT) grant SALGON (No. 01 IN 107).

References

[1] S. E. Fahlman. Faster–Learning Variations on Back–Propagation: An Empirical Study. Proceedings of the 1988 Connectionist Models Summer School. Morgan Kaufmann. 1988.

[2] S. E. Fahlman. Report, available on the FTP server "pt.cs.cmu.edu" in the directory "/afs/cs/project/connect/bench" of the Carnegie Mellon University. 1993.

[3] S. E. Fahlman, C. Lebiere. The Cascade–Corellation Learning Architecture. In Touretzky (ed.): Advances in Neural Information Pressing Systems 2. Morgan Kaufmann. 1990.

[4] F. Gruau. Genetic synthesis of boolean neural networks with a cell rewriting developmental process. Paper presented to ICANN 1992.

[5] F. Gruau. Cellular Encoding of Genetic Neural Networks. Report no. 92-21 Ecole Normale Superieure de Lyon, May 1992.

[6] K. J. Lang, M. J. Witbrock. Learning to Tell Two Spirals Apart, in Proceedings of the 1988 Connectionist Models Summer School. Morgan Kaufmann. 1988.

[7] H. Kitano. Designing neural networks using genetic algorithm with graph generation system. Complex Systems, 4:461–476, 1990.

[8] J. R. Koza. Genetic Programming. A paradigm for genetically breeding computer population of computer programs to solve problems. MIT press, 1992.

[9] A. Lindenmayer. Mathematical Models for Cellular Interaction in Development, Parts I and II. J. Theor. Biology. (18) 280–315, 1968.

[10] E. Mjolsness, D. H. Sharp, B.K. Alpert. Scaling, Machine Learning, and Genetic Neural Nets. Report. Yale University, YALEU/DCS/TR-613, March 1988.

[11] I. Rechenberg. Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. 1973. Frommann–Holzboog, Stuttgart.

[12] H.P. Schwefel. Numerical Optimization of Computer Models. J. Wiley, Chichester, New York, Brisbane, Toronto. 1981.

[13] D. E. Rummelhart, J. L. McClelland. Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Cambridge, Massachusetts, The MIT Press. 1986.

[14] H.–M. Voigt, J. Born, I. Santibáñez–Koref. Structuring of Artificial Neural Networks with Generative Grammars. Proceedings to the "Seminar Neuroinformatik 1992". Informatik-Fachberichte. Berlin, Springer-Verlag, 1993.