Final version appears in:
L. Kallel, B. Naudts, and A. Rogers, editors,
Theoretical Aspects of Evolutionary Computation, Springer-Verlag, 2000

Structure Optimization and Isomorphisms

P. Stagge and C. Igel

Insitut für Neuroinformatik
Ruhr-Universität Bochum
Bochum, Germany
E-mail: *peter.stagge@neuroinformatik.ruhr-uni-bochum.de*
E-mail: *christian.igel@neuroinformatik.ruhr-uni-bochum.de*

Abstract. In this article we deal with a quite general topic in evolutionary structure optimization, namely redundancy in the encoding due to isomorphic structures. This problem is well known in topology optimization of neural networks (NNs) and we study it in this framework. Choosing a good neural network structure for a given problem is still a difficult task for which we do not know any successful analytical means. But problem specific structures can lead to significantly improved results; evidence for this is given in this contribution by a NN that was evolutionary adapted for a benchmark problem. The degree to which isomorphic structures, i.e. classes of equivalent NN topologies, enlarge the search space depends on the restrictions of the allowed structures and on the representation of the search space. In the context of structure optimization of NNs we observe similar phenomena of rare and frequent structures as are known from molecular biology. To cope with isomorphisms we make use of the relation between NN topologies and graphs. Exploiting methods from graph theory we demonstrate a general way to deal with isomorphic structures. The presented approach applied to NN optimization can be regarded as a solution to the so called *competing conventions* problem [19]. Further, it can be used to save fitness evaluations when used in combination with a graph-database.

Keywords
structure and topology optimization, recurrent neural networks, isomorphism, graph representation, competing conventions problem, rare and frequent structures

1 Introduction

In this paper we present a way to deal with isomorphic structures in evolutionary structure optimization, see also Marino [10] and Van Hoyweghen [26] in this volume. Topology adaptation of artificial neural networks [1, 8] serves as an example.

When searching for a good NN with an evolutionary algorithm (EA) one has to code for the network structure; it can for example be described by its connection matrix. In general, there are several connection matrices leading to functionally equivalent network topologies i.e. isomorphic structures, see figure 1.

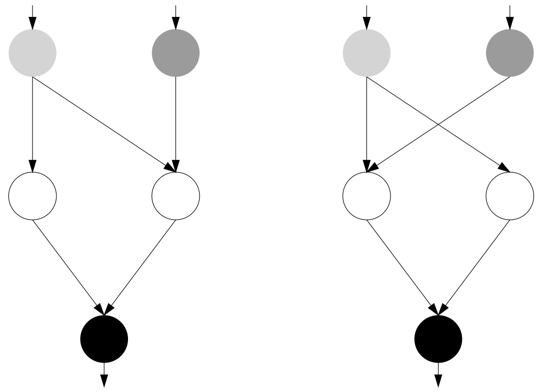

Fig. 1. Two isomorphic network structures. Each input neuron and the output neuron have different colours.

Such a redundant representation can be harmful or beneficial for evolutionary search processes. For example the mapping from RNA sequences to RNA secondary structures in biology is very well studied and proves to be utmost redundant. This redundancy is assumed to be helpful to avoid local optima by constituting *neutral networks* [20]. However, it seems to be a waste to evaluate isomorphic structures several times according to different connection matrices. Here, we present a way to circumvent this problem. We propose to construct graph representations that are free of isomorphic topologies and preserve neighborhood relations with respect to "canonical" mutations (addition or deletion of a node or connection).

The article begins with an introduction into structure optimization of NNs with EAs, then the concepts of isomorphic graphs, canonically labeled graphs, and rare and frequent structures are given. These are applied to artificial neural networks. The investigated benchmark example is described concisely. Finally, the results are given and a conclusion ends this article.

2 Evolutionary Structure Optimization of NNs

The motivation for optimizing the structure of NNs is the fact that we know about the approximation capability of NNs [9], but we do not know how a beneficial structure looks like for a given problem. Often we evaluate a NN according to its generalization error and the time it takes to train the network, but there are only very general hints that tell us how the structure of a neural network influences these values. The fact that standard structures are widely used (multi layer perceptrons with 1, 2 or 3 hidden layers, fully connected) does not reflect deeper insight into the theory of neural networks but demonstrates that there is no systematic and reliable way to generate problem specific network topologies. Results from structure optimization of NNs show that evolutionary algorithms are very suitable to optimize the topology of NNs [2, 19, 21, 27].

In order to tackle the structure optimization problem in the framework of EAs one has to define a genotype space which constitutes the search space, and search operators e.g. mutation or crossover. Each genotype in the search space can be mapped to a NN (simple example: connection matrix as genotype being mapped to a NN). Representation and operators strongly depend on one another.

In this work we consider only evolutionary optimization of network topologies. Neither the weights nor weight initialisations are included in the genotype. Each NN structure has to be trained from a random weight initialisation and the resulting weights are not coded back into the genotype. There are a lot of different approaches using different codings and operators which can be crudely divided into two categories:

In *direct encoding* each connection in the neural network is explicitly represented in the genotype e.g. in a connection matrix [14]. Thus, one can get a good idea of the effects the search operators on the genotypes will have in the structure space e.g. connection deletion.

In *indirect encoding* schemes the genotype does not represent the NN topology directly, but it consists of – for example grammar based – rules that describe the construction process for the NN. The mapping from a genotype to a NN structure might even be non-deterministic, in some cases connections are given according to a coded connection density. There are a lot of different indirect encoding schemes some of which are biologically inspired as models for the natural neural growth processes, for an overview see [21].

Any approach has to cope with several problems. There might be structures which are represented by numerous genotypes, others only by few or one genotype, and in indirect encoding schemes some might not be coded for at all. Search operators can have the tendency to produce NNs with specific characteristics (bias in structure space) e.g. NNs with many layers or sparsely connected NNs. The influence of the operators on the structure and on the fitness is not obvious. With regard to EAs one asks for operators obeying the principle of "strong causality" [16] i.e. a small change in the genotype causes a small change in the fitness value (the distance in genotype space should be defined by the search operators [22]). Up to now there is neither a proven best representation–operator setting nor a constructive way to generate a good setting for a given problem except some general design heuristics.

3 NNs and Graph Isomorphisms

3.1 Graph Isomorphism

In this section, we summarize some classical graph theory needed to identify isomorphic neural networks. The notation and results used in this section are based on McKay[11, 12].

Let $G = (V, E)$ be a directed graph, V and E denote the set of vertices and edges, respectively. A *colouring* π denotes an ordered partition of V into disjoint, non-empty colour classes i.e. for k different colours $\pi = (V_1, V_2, \ldots, V_k)$. The order of the colour classes is significant, but the order of the vertices within each class is not.

Let γ be a permutation of the vertex set V of a graph G and let v^γ denote the image of the vertex $v \in V$ under γ. Similar, we define $W^\gamma = \{w^\gamma | w \in W\}$ with $W \subseteq V$ and G^γ as the graph in which vertices v^γ and w^γ are adjacent iff v and w are adjacent vertices in G. The image of a colouring π under γ is defined as $\pi^\gamma = (V_1^\gamma, V_2^\gamma, \ldots, V_k^\gamma)$. A permutation γ of a vertex set with colouring π is called *colour-preserving* iff $\pi^\gamma = \pi$ i.e. each $v \in V$ has the same colour as v^γ. Two coloured graphs G_1 and G_2 are isomorph iff there exists a colour-preserving permutation γ that transforms one into the other, say $G_1 = G_2^\gamma$.

Let $\pi = (V_1, V_2, \ldots, V_k)$ be a partition of $\{0, 1, \ldots, n-1\}$, then we define $c(\pi)$ as the partition $(\{0, 1, \ldots, |V_1| - 1\}, \{|V_1|, \ldots, |V_1| + |V_2| - 1\}, \ldots, \{n - |V_k|, \ldots, n-1\})$ i.e. $c(\pi)$ is independent of π except it has the same number of classes with the same sizes in the same order.

A *canonical labelling map* is a function $\mathcal{C}$ such that for any graph G, partition π of V, and permutation γ of V we have

1. $\mathcal{C}(G, \pi) = G^\delta$ for some permutation δ such that $\pi^\delta = c(\pi)$, and
2. $\mathcal{C}(G^\gamma, \pi^\gamma) = \mathcal{C}(G, \pi)$.

The usefulness of canonical labelling maps is shown by the following theorem [12].

Theorem 1 (McKay 1990). *Suppose the graphs G_1 and G_2 are coloured using the same number of vertices of each colour. Then $\mathcal{C}(G_1, \pi_1) = \mathcal{C}(G_2, \pi_2)$ iff $G_1^\gamma = G_2$ for some colour-preserving permutation γ. (Here, π_1 and π_2 are the colourings, with the colours in the same order in each.)*

The general graph-isomorphism problem and therefore the canonical labelling is a potential member of the problem class NPI which contains the so called NP-incomplete problems. NPI is defined as NP excluding P and all NP-complete problems [6]. So we cannot expect to have an algorithm which solves the general graph-isomorphism problem in polynomial time. However, efficient heuristic algorithms exist that can handle the canonical labelling for graphs generated from neural networks. The graphs representing neural network which can be regarded as tractable with todays learning algorithms have moderate size, so that the costs of computing their canonical representation can almost be neglected compared to the fitness evaluation (learning) time.

Fast[1], well documented software that implements a canonical labeling map is freely available: the *nauty* package by Brendan D. McKay is used in this investigation.

3.2 NNs as Graphs

Feed-Forward NNs We want to use graph theory to determine whether two NN topologies are isomorph. Therefore, we need a way to map a network to a (coloured) graph.

[1] Computing the canonically labeled graph representing a neural network consisting of 100 nodes takes in the order of 10^{-2}s on a 250MHz UltraSparc.

Suppose a feed-forward network $\mathcal{N}$ with n neurons $x_0, \ldots, x_{n-1}$. Without loss of generality, we assume that the neurons $x_0, \ldots, x_{d-1}$ are the d input neurons, $x_{n-m}, \ldots, x_{n-1}$ are the m output neurons, and that each neuron x_i gets only input from neurons x_j with $j < i$. The output y_i of each neuron i is then given by

$$y_i = g\left(\sum_{j=0}^{i-1} w_{ij}^0 y_j + \vartheta_i\right) \quad , \tag{1}$$

where g is a so called activation function, ϑ_i is the threshold (bias), and w_{ij}^0 is the strength (weight) of the connection between neuron j and i.

The vertex set of the corresponding directed graph $G_{\mathcal{N}} = (V_{\mathcal{N}}, E_{\mathcal{N}}^0)$ is $\{0, \ldots, n-1\}$ and there is an edge between two vertices u and v iff there is a connection from neuron x_u to neuron x_v. As $\mathcal{N}$ is a feed-forward network, $G_{\mathcal{N}}$ is acyclic. We use a coloured graph to distinguish between different classes of neurons, e.g. neurons that are connected to certain inputs or outputs. We define the colouring

$$\begin{aligned}\pi_{\mathcal{N}} = (\{0\}, \{1\}, \ldots, \{d-1\}, \{d, d+1, \ldots, n-m-1\}, \\ \{n-m\}, \{n-m+1\}, \ldots, \{n-1\})\end{aligned} \tag{2}$$

to fix the input and output units, see figure 1 for an example.

Recurrent NNs The recurrent networks considered in this investigation here are generalized Elman networks [3, 24, 25]. Let $y_i(t)$ denote the output of neuron i at time t. A network with S memory layers obeys the following dynamics

$$y_i(t) = g\left(\sum_{j=0}^{i-1} w_{ij}^0 y_j(t) + \sum_{s=1}^{S}\sum_{j=0}^{n-1} w_{ij}^s y_j(t-\tau_s) + \vartheta_i\right) \quad . \tag{3}$$

The w_{ij}^s for $s > 1$ are the weights for recurrent connections from memory layer s, and τ_s is the corresponding time-delay, see [24, 25] for details. The structure of such a network $\mathcal{N}$ can be described by a set of vertices $V_{\mathcal{N}} = \{0, \ldots, n-1\}$ and $S+1$ sets of edges, $E^s, s \in \{0, \ldots, S\}$. $E_{\mathcal{N}}^0$ is the same as for the feed-forward network. Additionally, there is an edge (u, v) in $E_{\mathcal{N}}^s$ iff there is a connection from x_u to x_v in memory layer s. For $s > 0$ there may be edges $(u, v) \in E_{\mathcal{N}}^s$ with $u \geq v$.

3.3 Detecting Isomorphic NNs

Feed-Forward NNs We want to use theorem 1 to determine whether the topologies of two NNs as defined in section 3.2 are isomorph.

The structure of two neural networks $\mathcal{N}_1$ and $\mathcal{N}_2$ is said to be isomorph iff $V_{\mathcal{N}_1} = V_{\mathcal{N}_2}$ and there is a bijection $\gamma : V_{\mathcal{N}_1} \to V_{\mathcal{N}_1}$ so that for each s, $0 \leq s \leq S$

$$(u, v) \in E_{\mathcal{N}_1}^s \leftrightarrow (\gamma(u), \gamma(v)) \in E_{\mathcal{N}_2}^s \quad . \tag{4}$$

Further, we require that input and output units of one network are mapped to the corresponding units of the other.

We can prove the following statement.

Proposition 1. *Two feed-forward neural network structures $\mathcal{N}_1$ and $\mathcal{N}_2$ are isomorph iff the coloured graphs $G_{\mathcal{N}_1}$ and $G_{\mathcal{N}_2}$ with colouring $\pi_{\mathcal{N}_1}$ and $\pi_{\mathcal{N}_2}$, respectively, are isomorph.*

In particular, the networks are isomorph iff

$$\mathcal{C}(G_{\mathcal{N}_1}, \pi_{\mathcal{N}_1}) = \mathcal{C}(G_{\mathcal{N}_2}, \pi_{\mathcal{N}_2}) \tag{5}$$

for some canonical labelling map $\mathcal{C}$. For a fixed function $\mathcal{C}$, for each feed-forward neural network $\mathcal{N}$ there exists exactly one isomorph *canonical network* which is uniquely defined by the graph $\mathcal{C}(G_{\mathcal{N}}, \pi_{\mathcal{N}})$.

The colouring ensures that input and output units of one network are mapped to the corresponding units of the other network. Further, if different types of neurons are encoded, the colouring can be used to group these neurons. For example, when a *sigma-pi* network [18] is described, colouring can prevent from mapping a sigma unit to a pi unit and vice versa.

Recurrent NNs In order to use theorem 1 for recurrent neural networks, we have to represent the network by a single directed coloured graph.

Let $\phi(.)$ be an injective mapping from the space of Elman network structures into the space of directed coloured graphs.

Proposition 2. *Two Elman topologies $\mathcal{N}_1$ and $\mathcal{N}_2$ are isomorph iff there is a colour-preserving isomorphism from $\phi(\mathcal{N}_1)$ onto $\phi(\mathcal{N}_2)$.*

Let $(V_{\phi,\mathcal{N}}, E_{\phi,\mathcal{N}}, \pi_{\phi,\mathcal{N}})$ be the coloured directed graph generated by $\phi(.)$ given an Elman network $\mathcal{N}$. If δ is a colour-preserving isomorphism from $\phi(\mathcal{N}_1)$ onto $\phi(\mathcal{N}_2)$, then $\gamma = \phi^{-1} \circ \delta \circ \phi$ is a colour-preserving isomorphism from $\mathcal{N}_1$ onto $\mathcal{N}_2$.

$$\begin{array}{ccc}
\mathcal{N}_1 & \xrightarrow{\phi} & (V_{\phi,\mathcal{N}_1}, E_{\phi,\mathcal{N}_1}, \pi_{\phi,\mathcal{N}_1}) \\
\gamma \downarrow & & \downarrow \delta \\
\mathcal{N}_2 & \xleftarrow{\phi^{-1}} & (V_{\phi,\mathcal{N}_2}, E_{\phi,\mathcal{N}_2}, \pi_{\phi,\mathcal{N}_2})
\end{array}$$

In the following, we give one possible choice for ϕ. Suppose an Elman topology $\mathcal{N}$ with n neurons and S memory layers. The injective mapping $\phi(.)$ generates a coloured directed graph with $(S+1)n$ vertices. The set of edges is constructed the following way.

Starting from $E_{\phi,\mathcal{N}} = \emptyset$

$$\begin{aligned} \forall s = 0, \ldots, S \;\; \forall [x, y] \in E^s_{\mathcal{N}} : \quad & E_{\phi,\mathcal{N}} = E_{\phi,\mathcal{N}} \cup \{ [ns + x, ns + y] \} \\ \forall x \in V_{\mathcal{N}} : \quad & E_{\phi,\mathcal{N}} = E_{\phi,\mathcal{N}} \cup \bigcup_{s=1}^{n} \{ [x, ns + x] \} \;. \end{aligned}$$

The colouring $\pi_{\phi,\mathcal{N}}$ is used to distinguish between vertices that "represent" different memory layers. As in (2), the input and output nodes are assigned different colours

$$\begin{aligned} \pi_{\phi,\mathcal{N}} = {} & (\{0\}, \{1\}, \ldots, \{d-1\}, \{d, d+1, \ldots, n-m-1\}, \\ & \{n-m\}, \{n-m+1\}, \ldots, \{n-1\}, \\ & \{n\}, \{n+1\}, \ldots, \{n+d-1\}, \{n+d, n+d+1, \ldots, 2n-m-1\}, \\ & \{2n-m\}, \{2n-m+1\}, \ldots, \{2n-1\}, \\ & \{2n\}, \{2n+1\}, \ldots, \{2n+d-1\}, \{2n+d, d+1, \ldots, 3n-m-1\}, \\ & \{3n-m\}, \{3n-m+1\}, \ldots, \{3n-1\}, \ldots, \\ & \{Sn\}, \{Sn+1\}, \ldots, \{Sn+d-1\}, \{Sn+d, d+1, \ldots, (S+1)n-m-1\}, \\ & \qquad \{(S+1)n-m\}, \{(S+1)n-m+1\}, \ldots, \{(S+1)n-1\}) \;. \qquad (6) \end{aligned}$$

4 Frequent and Rare Structures

Having defined isomorphic networks, a simple example shall give an idea about the number of equivalent NNs. Consider a feed-forward NN with d input nodes, l hidden nodes in one hidden layer i.e. they are only connected to the inputs and the output, and 1 output node – a d-l-1 NN. The connections between the hidden nodes and the output node are fixed. For the sake of simplicity, we allow hidden neurons to have no connection to the input. In an encoding with a connection matrix, there is a field of $d \cdot l$ bits specifying the connections between input and hidden neurons. Thus, there are $G = 2^{d \cdot l}$ different matrices in the genotype space. The following reasoning gives the number of different structures. For each hidden neuron the connection to the input neurons is given by a bitstring of size d, yielding $T = 2^d$ different possible types for a hidden node. For a d-l-1 NN the multi-set of the types of the hidden nodes, $\{t_1, \cdots, t_l\}$, with $t_i \in \{0, \cdots, T-1\}$ specifies a class of equivalent NNs. Combinatorics gives an expression for the number of these different sets, in our case for the number of not equivalent NNs

$$S = \binom{T + l - 1}{l} \;.$$

We can define the average number of matrices per NN class as: $\bar{R} = G/S$. For example, $\bar{R} = 67.6326$ for a 4-5-1 NN.

There are equivalent NNs for which a large number of matrices code and others for which only few matrices or even just one matrix codes. To get an impression how often equivalent NNs are represented in the genotype space, let us define *frequent* and *rare structures* [7].

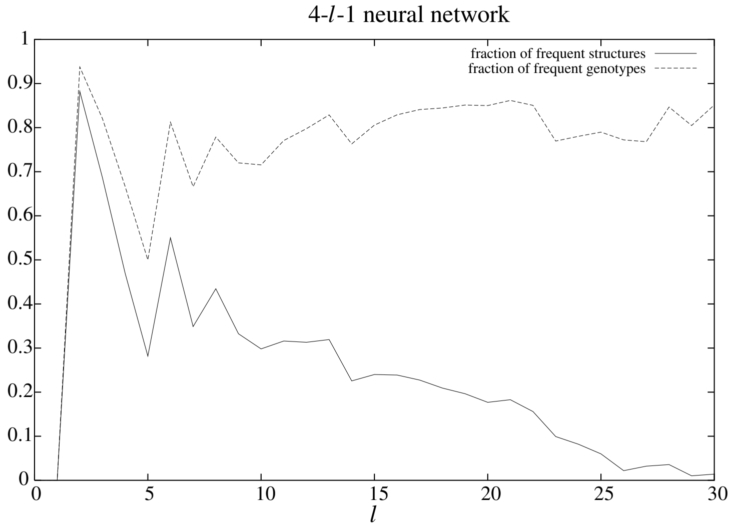

Fig. 2. Fraction of *frequent* structures (solid line) and of matrices coding for these structures (dashed line) as the number of the hidden neurons increases.

Definition 1. *An equivalence class (a structure) is frequent iff the number of matrices generating a network of this class is above average $\bar{R}$. Else it is rare.*

For an increasing number of hidden nodes l, the following observation is made. The fraction of frequent structures decreases, whereas the fraction of matrices that code for a frequent structure increases, see figure 2. That is the majority of genotypes – matrices – codes for a minority of structures – equivalent NNs. Similar considerations may hold for other structure optimization problems too. An encoding that is based on equivalent classes can circumvent this artefact.

5 Experiments

5.1 Evolutionary Algorithm

For the simulations presented here we employ a direct encoding and a small number of intuitive operators.

In genotype space an individual consists of a connection matrix; additionally, each neuron belongs to a layer. The NNs are not allowed to have more than a maximum number of layers. All feed-forward connections including all short-cuts are allowed as long as the layer constraint is met.

One of the used operators is to split a layer into two layers; this does not change the topology of the network but it allows new connections that were not allowed before.

There is also an operator that merges two successive layers. This may delete connections between neurons in the originally separated layers. We also use an operator that adds or removes a connection. Adding a new connection increases the capabilities of the NN but might make learning more difficult and can lead to over-fitting. Deleting a connection decreases the capabilities of the NN but can benefit the learning process. Thus, the effect of these operators is not easily assessed. Furthermore, there is a node insertion operator. The new node will get 2 inputs and 1 output. The output is directed into the output layer, the input is received from two randomly chosen preceding nodes. Finally, there is a node deletion operator. As this causes a great structural change, those nodes which received input from the deleted node will get new connections from those nodes that provided the input to the deleted node. So, this operator possibly introduces a lot of new connections by deleting a node. We ensure that the operators only produce feasible networks.

One effect of our coding obviously is the restriction in the number of layers. This is not only a topological constraint, which relies on the idea that layers are meaningful to the adaptation process of NNs, but it also reduces the search space. As there are much more NNs with many layers we also reduce in this way a possible bias towards deep structures: Assume that a connection is set with a probability p, then the probability that n subsequent neurons belong to one layer is

$$P(n) = (1-p)^{\frac{n(n-1)}{2}} \quad . \tag{7}$$

This shows that large layers become very rare which in turn leads to deep structures.

The used EA consists of a parent population with 15 individuals which produce an offspring population with 90 individuals, by randomly selecting one parental genotype and applying one of the introduced operators drawn according to different probabilities until a topological change occurs. Thereafter, each offspring is evaluated i.e. the NN is trained. If the NN has been trained before during the course of running the EA, the formerly computed fitness will be used. The parent population for the next generation is determined in the following way: Out of the old parents and the offsprings the very best individual is placed in the new parental population i.e. we keep one elitist. The other 14 individuals are deterministically selected from the offspring population according to their rank.

5.2 Test Problem

The example data set called *diabetes1* is taken from the PROBEN1 [15] benchmark collection. The goal of this real-world classification task is to decide whether a Pima Indian individual is diabetes positive or not. There are eight inputs and a 1-of-2 encoding is used for the output. The size of the training, validation and test set is 384, 192 and 192, respectively. Some of the patterns are disturbed by noise; they contain senseless zero entries.

Following [15], the quality of the evolved networks was determined by the error percentage[2]. The resilient backpropagation (RPROP) algorithm [17] was used to train the nets. The RPROP parameters were $\eta^+ = 1.1$, $\eta^- = 0.5$, $\Delta_0 = 0.01$, $\Delta_{\min} = 0$ and $\Delta_{\max} = 50.0$. The initial weights were chosen randomly between -0.5 and 0.5. The hidden units used sigmoidal activation function given by $g(x) = x/(1+|x|)$ and the output units were linear. All nets were trained for 300 iterations. The weight configuration with the lowest error on the validation set was stored. In this experiment, we wanted to force adaptation to the test set, so the error of the network on the test set determined the fitness[3]. Of course, normally the test set must not be used in the model adaptation process.

In [15], 12 different standard network architectures and different initialisations were tested. The networks had one or two hidden layers and up to 32 hidden units. Each architecture was tested with sigmoidal and linear output units. The best network (using the *diabetes1* data set) found had a test error of 16.47 and a classification error of 25%. The same problem was used as a benchmark in [13] for 23 different classification techniques, where the best algorithm achieved a classification error of 22.3%.

5.3 Graph Database

During the evolutionary process each generated network was canonically labeled and this canonically labeled graph was stored in a database. In advance it was tested whether this network or an isomorphic one was evaluated before by looking up the canonically labeled graph in the database. Such a database can be used to reduce the number of computational expensive fitness evaluations. This idea was firstmentioned in the work of Christoph M. Friedrich [4, 5] but no experiments were performed.

5.4 Results

We conducted two runs of the algorithm described in section 5.1 on the diabetes problem. A maximum of 4 hidden layers was allowed.

The first run found a network with 49 connections and a classification error on the test set of 19.27% in generation 43. The error on the test set dropped to 15.40 in generation 200. The second run evolved a network with 49 connections and a minimum test error of 15.22 in generation 95. The classification error of this network on the test set was 19.79%. We performed a single additional run with a slightly changed selection scheme and found a net with a classification error of 18.75% and 51 connections. The results can not be directly compared to those in the literature [13, 15] as we used the test data set to calculate the fitness. So we do not get a true generalization error. Moreover, training

[2] The error percentage is defined as $E = \frac{100}{N \cdot d \cdot (t_{\max} - t_{\min})^2} \sum_{n=1}^{N} (\boldsymbol{o}_n - \boldsymbol{t}_n)^2$, $\boldsymbol{t}$: target, $\boldsymbol{o}$: net output, N: size of training data set, $t_{\min}(t_{\max})$: minimum (maximum) target value.

[3] This was done in order to show how different statistics of test and validation data influence the topology optimization process, see also section 5.4.

a standard architecture on the training data and monitoring validation and test error reveals that these data sets contain quite different statistics. Thus, including the test error in the fitness value may lead to an overfitting of the network structure to that data set. Nevertheless, the experiments demonstrate that the proposed method is an efficient and robust method to determine neural network topologies.

In the runs of the algorithm outlined in section 5.1 the average number of isomorphic networks detected per generation was 2.22 and 2.75, respectively i.e. more than 2.75% of the evaluations could be saved.

6 The Competing Conventions Problem

One point that has been addressed by several researchers in the field of structure optimization of NN is the *competing conventions* problem [19]. That is, many genotypes (conventions) map to equivalent structures i.e. the corresponding network topologies are isomorph. It was argued that this has a harmful effect on recombination, "Crossover among parents utilizing the same convention are likely to be successful while crossovers between dissimilar conventions are unlikely to be effective" [19]. Our results show that there are not so many isomorphic nets present in one generation as might be expected (for the population size and selection mechanism used in our investigation). So the probability of selecting two competing conventions of one net for recombination is rather small. Further, if only canonically labeled networks are stored in the population, there is exactly one representation of each network and the competing conventions problem is solved (at least if the weights are not encoded into the genotype). This requires that each genotype which is created by the search operators is replaced by the genotype belonging to the canonical representative of the network structure.

However, we do not believe that crossover on the genotype level is an important operator for the evolution of neural networks using a direct encoding, therefore it has been omitted in our algorithm. The isomorphism-free representation ensures a single "convention", but does not necessarily lead to something like a "homologous" crossover between NNs.

7 Discussion

In structure optimization problems the space of structures is given by the problem, but for an EA the genotype space and the search operators are chosen according to design criteria for successful evolutionary search processes. This can lead to a genotype space that is much larger than the number of different structures. In these cases it seems very reasonable to organize the fitness evaluation process in an isomorphism-free space. We exemplified this procedure in a simple but realistic problem. The chosen setup did not try to enforce isomorphic network structures. The number of isomorphic structures increases if one reduces the number of allowed hidden layers and if one forbids shortcut connections. Additionally, different choices of the selection procedure can possibly lead to a more localized search yielding a higher rate of already evaluated network structures. Furthermore, our results show that NN structures which are adapted to the problem can lead to much better results than those standard structures achieve.

Keeping an elitist with its fitness in an EA with a noisy fitness function – in our case the learning process is noisy due to the random initialization – might lead to a good individual with an exceptionally good learning result blocking further developments. In this case, one can train the NN several times, possibly in the next generations, until there is a reliable statistics. This and further refinements concerning the selection process on noisy fitness functions [23] work very well together with the proposed method of using canonical labeled graphs.

Summarizing our investigation, we have shown that:

- The phenomena of rare and frequent structures exists in technical real-world applications.
- Smart techniques to deal with isomorphic structures can be beneficial in evolutionary structure optimization.
- The labelled, coloured graph isomorphism question can be reduced to a coloured graph isomorphism problem.
- The competing conventions problem can be circumvented in the field of NN topology adaptation.

Acknowledgments

C. Igel thanks C. M. Friedrich for fruitful discussions and acknowledges support from the BMBF under grant LEONET 01IB802A9.

References

1. C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 1995.
2. H. Braun. *Neuronale Netze: Optimierung durch Lernen und Evolution*. Springer Verlag, 1997.
3. J. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.
4. C. M. Friedrich. Entwicklung und Analyse Evolutionärer Algorithmen zur Optimierung der Struktur und Parameter von künstlichen Neuronalen Netzen. Master's thesis, Universität Dortmund, Fachbereich Informatik, 1995.
5. C. M. Friedrich and C. Moraga. An evolutionary method to find good building-blocks for architectures of artificial neural networks. In *Sixth International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU '96)*, pages 951–956, Granada, Spain, 1996.
6. M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

7. W. Grüner, R. Giegerich, D. Strothmann, C. Reidys, J. Weber, I. Hofacker, P. Stadler, and P. Schuster. Analysis of RNA sequence structure maps by exhaustive enumeration II. Structures of neutral networks and shape space covering. *Monatshefte für Chemie Chemical Monthly*, 127:375–389, 1996.
8. S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, New Jersey, 2nd edition, 1998.
9. K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359 – 366, 1989.
10. A. Marino and A. Prügel-Bennett. Genetic computation and highly symmetric problems. In L. Kallel, B. Naudts, and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computation*, Natural Computing. Springer-Verlag, 2000.
11. B. D. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30(30):47–87, 1981.
12. B. D. McKay. *nauty* user's guide (version 1.5). Technical Report TR-CS-90-02, Australian National University, Computer Science Department, 1990.
13. D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.
14. G. Miller and P. Todd. Designing neural networks using genetic algorithms. In J. Schaffer, editor, *3rd Int Conference on Genetic Algorithms and their Applications*, pages 379–384. Morgan Kauffman, 1989.
15. L. Prechelt. PROBEN1 — A set of benchmarks and benchmarking rules for neural network training algorithms. Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, 1994.
16. I. Rechenberg. *Evolutionsstrategie '94*. Friedrich Frommann Holzboog Verlag, Stuttgart, 1994.
17. M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*. IEEE Press, 1993.
18. D. E. Rummelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error backpropagation. In D. E. Rummelhart, J. L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–362. MIT Press, Cambridge, MA, 1986.
19. J. D. Schaffer, D. Whitley, and L. J. Eshelman. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In L. D. Whitly and J. D. Schaffer, editors, *COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pages 1–37, 1992.
20. P. Schuster and W. Fontana. Continuity in evolution: On the nature of transitions. *Science*, 1998.
21. B. Sendhoff. *Evolution of Structures*. PhD thesis, Institut für Neuroinformatik, Ruhr–Universität Bochum, 1998.
22. B. Sendhoff, M. Kreutz, and W. von Seelen. A condition for the genotype–phenotype mapping: Causality. In T. Bäck, editor, *Genetic Algoritms, Proceedings of the 7th International Conference (ICGA)*, pages 73–80. Morgan Kaufmann, 1997.
23. P. Stagge. Averaging efficiently in the presence of noise. In A. Eiben, editor, *Parallel Problem Solving from Nature, PPSN V*, Lecture Notes in Computer Science, pages 188–197. Springer, 1998.
24. P. Stagge and B. Sendhoff. An extended Elman net for modeling time series. In W. Gerstner, editor, *International Conference on Artificial Neural Networks*, Lecture Notes in Computer Science. Springer, 1997.

25. P. Stagge and B. Sendhoff. Organisation of past states in recurrent neural networks: Implicit embedding. In M. Mohammadian, editor, *Computational Intelligence for Modelling, Control & Automatation*, pages 21–27, 1999.
26. C. Van Hoyweghen. Detecting spin-flip symmetry in optimization problems. In L. Kallel, B. Naudts, and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computation*, Natural Computing. Springer-Verlag, 2000.
27. X. Yao. Evolving artificial neural networks. *Proc. of the IEEE*, 87(9):1423–1447, 1999.