Neural Network Structures and Isomorphisms: Random Walk Characteristics of the Search Space

Peter Stagge **Christian Igel**
Insitut für Neuroinformatik
44780 Ruhr-Universität Bochum, Germany
Phone: +49 234 32 25558
{Peter.Stagge,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).

In the context of structure optimization of NNs we observe similar phenomena of rare and frequent structures as are known from molecular biology. 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. For restricted network topologies, like NNs with a maximum number of layers, some properties can be analyzed analytically. For more general structures we estimate the characteristics of the search space using data stemming from random walks.

For restricted NN topologies, the search process is affected by isomorphic structures. However, in the absence of restrictions, the search space becomes so large that the bias induced by isomorphisms can be neglected.

1 Introduction

There is not much theory on structure optimization, neither in general nor for neural network architectures in particular. In this article we propose an approach to analyze redundancy in the search space of graph representations and present some preliminary results for neural network (NN) structure optimization.

To set up an efficient optimization process it is essential to analyze the search space and to take its characteristics into consideration. They determine the computational cost of the search and—at least partly—the final outcome. Defining *structure* as the arrangement of parts of a whole to one another [13] then graph representations become obvious for structure optimization tasks. Isomorphic graphs constitute the source for redundancy in this encoding. Therefore, this article contributes to the question how isomorphic graph topologies influence the search space. However, we believe that redundancy is a typical property of structure encoding and not restricted to graph representations.

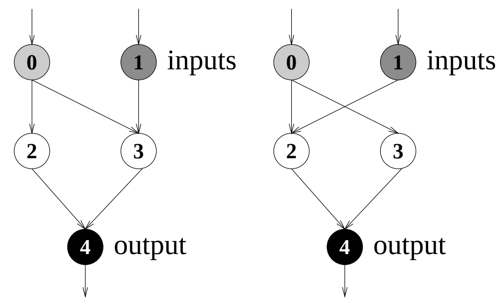

Figure 1: Two isomorphic 2-2-1 network structures; each input neuron and the output neuron have different colors.

In this work, we consider evolutionary optimization of NN topologies, see [13, 15] for surveys of this field. When searching for a good NN structure with an evolutionary algorithm (EA) one has to code for the network topology, which can for example be described by its connection matrix, that is the adjacency matrix of the corresponding network graph. In general, there are several connection matrices leading to functionally equivalent network topologies, i.e., to isomorphic structures, see Fig. 1. In [14] we presented a way to deal with isomorphic structures in evolutionary structure optimization, in particular in the context of topology optimization of feed-forward and recurrent neural networks. We assume a *direct encoding* where each connection in the neural network is explicitly represented in the genotype. Thus, one can get a good idea of the effects the search operators acting on genotypes will have in the structure space, e.g., connection deletion. Neither the weights nor weight initializations are included in the genotype. Further, we require that there is an upper bound on the number of nodes, i.e., the search space is finite.

A redundant representation can be harmful as well as 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* [12]. In this article we will discuss some of the harmful aspects of a redundant NN representation. Irrespective of whether the redundancy turns out to be beneficial or not, it seems to be a waste to evaluate isomorphic topologies several times due to different genotypes.

In the next section we review the relation between NNs and graph isomorphisms, the usefulness of canonical labeling maps, and a way to calculate the number of different adjacency (connection) matrices that code for isomorphic structures. In Sec. 3 we introduce some concepts to describe redundant search spaces and analyze certain NN search spaces both analytically and empirically. Section 4 briefly summarizes possible applications of the proposed methods. A conclusion ends this article.

2 NNs and Graph Isomorphisms

In this section, we summarize some classical graph theory needed to identify isomorphic neural networks. The notation and results are mainly based on [8, 9]. As we analyze the redundancy of direct encodings the number of adjacency matrices that represent topologies that are isomorph to a given graph is asked for.

2.1 Graph Isomorphism

Let $G = (V, E)$ be a directed graph, V and E denote the set of vertices and edges, respectively. A *coloring* π denotes an ordered partition of V into disjoint, non-empty subsets called color classes, i.e., $\pi = (V_1, V_2, \ldots, V_k)$ for k different colors. The order of the color classes is significant, but the order of the vertices within each class is not. Two graphs G_1 and G_2 with colorings π_1 and π_2 are isomorph iff there exists a color-preserving permutation of the vertices that transforms one graph into the other. Color-preserving means that the permutation maps each vertex onto a vertex with the same color, see [8, 9, 14] for details.

A set of graphs can be divided into equivalence classes of isomorphic topologies. One can construct functions called *canonical labeling maps* that map every graph to one unique (canonical) representative of its equivalence class. Thus, for a canonical labeling map $\mathcal{C}$ it holds:

Theorem 1 (McKay 1990). *Suppose the graphs G_1 and G_2 are coloured using the same number of vertices of each color. Then $\mathcal{C}(G_1, \pi_1) = \mathcal{C}(G_2, \pi_2)$ iff G_1 and G_2 are isomorph. (Here, π_1 and π_2 are the colorings, with the colors in the same order in each.)*

2.2 Counting adjacency matrices

Given a graph $G = (V, E)$, we want to answer the question how many different adjacency matrices exist that code for graphs that are isomorph to G. Neglecting the coloring, there are $|V|!$ permutations of the vertices. Now suppose that G is colored and that there are k different color classes. Then there exist $|V_1|! \cdot |V_2|! \cdot \ldots \cdot |V_k|!$ color-preserving permutations, but some of these permutations do lead to the same adjacency matrices. Two permutations give identical adjacency matrices iff there is an *automorphism mapping* from one of the permutations to the other. Hence, we need to know the number of such automorphisms.

Definition 1. *The* size of the automorphism group *of a graph G denoted by $a(G)$ is defined as the number of color-preserving permutations of the vertices that map G onto itself, that is the number of automorphisms.*

For example, let G be one of the graphs in Fig. 1. Then $a(G) = 1$ because the identity is the only color-preserving permutations that maps G onto itself.

Let π_1 be a color-preserving permutation and π_2 an automorphism. Then the concatenation $\pi_1 \circ \pi_2$ of the two permutations leads to the same adjacency matrix as the permutation π_1. As $a(G)$ is the number of automorphisms, there exist $a(G)$ permutations for each permutation that lead to the same adjacency matrix. Hence, the product of factorials derived above has to be normalized by $a(G)$ to account for the automorphisms:

Proposition 1. *The number of different adjacency matrices that code for graphs that are isomorph to G is given by*

$$\frac{1}{a(G)} \prod_{i=1}^{k} |V_i|! \ . \tag{1}$$

For example, there are three different adjacency matrices that code for the networks that are isomorph to the one shown in Fig. 2. The size of the automorphism group is two (the identity and the permutation that swaps the nodes 3 and 4), therefore we have $1 \cdot 1 \cdot 3! \cdot 1/2 = 3$.

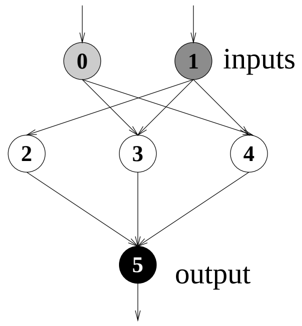

Figure 2: There are three different adjacency matrices that code for networks that are isomorph to this one.

2.3 Feed-forward NNs as Graphs

To use graph theory to determine whether two NN topologies are isomorphic or not, we need a way to map a network to a (colored) graph. We restrict our considerations to feed-forward NNs, see [14] for the treatment of recurrent neural networks. 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 and $x_{n-m}, \ldots, x_{n-1}$ are the m output neurons.

The vertex set $V_{\mathcal{N}}$ of the corresponding directed graph $G_{\mathcal{N}} = (V_{\mathcal{N}}, E_{\mathcal{N}})$ is $\{0, \ldots, n-1\}$; there exists 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. A colored graph is used to distinguish between different classes of neurons, e.g., neurons that are connected to certain inputs or outputs. The coloring

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

can be used to fix the input and output units, see Fig. 1 for an example.

The topologies of two neural networks $\mathcal{N}_1$ and $\mathcal{N}_2$ are said to be isomorphic iff $V_{\mathcal{N}_1} = V_{\mathcal{N}_2}$ and there is a bijection $\gamma : V_{\mathcal{N}_1} \to V_{\mathcal{N}_1}$ such that

$$(u, v) \in E_{\mathcal{N}_1} \leftrightarrow (\gamma(u), \gamma(v)) \in E_{\mathcal{N}_2} \ . \quad (3)$$

Further, we require that input and output units of one network are mapped to the corresponding units of the other. Based on Theorem 1 it holds:

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

In particular, the networks are isomorph iff

$$\mathcal{C}(G_{\mathcal{N}_1}, \pi_{\mathcal{N}_1}) = \mathcal{C}(G_{\mathcal{N}_2}, \pi_{\mathcal{N}_2}) \quad (4)$$

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

The coloring 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 coloring can be used to group these neurons. For example, when a *sigma-pi* network [10] is described, coloring can prevent from mapping a sigma unit to a pi unit and vice versa.

2.4 Complexity

The general graph isomorphism problem and therefore the canonical labeling 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 [5]. 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 labeling for graphs generated from NNs. The graphs representing NNs 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 and determines the size of the automorphism group is freely available: the *nauty* package [9] by Brendan D. McKay is used in this investigation.

3 Analysis of the Search Space

In this section, we introduce the notion of rare and frequent structures and present some analytical and empirical analysis of the redundancy in the NN topology search space.

3.1 Frequent and Rare Structures

Let $\mathcal{G}$ denote the genotype space. In a direct encoding, $\mathcal{G}$ corresponds to the matrices that code for a feasible NN topology and additionally may satisfy certain constraints. Let $\mathcal{S}$ be the set of equivalence classes of neural network structures that are covered by $\mathcal{G}$. The surjective function $\Gamma : \mathcal{G} \to \mathcal{S}$ maps each genotype to the corresponding equivalence class. For example, both network topologies in Fig. 1 are mapped to the same element of $\mathcal{S}$. As the elements in one equivalence class are functionally equivalent neural networks, we refer to the elements of $\mathcal{S}$ as structures throughout the remainder of this article.

Now we define the redundancy of an equivalence class:

Definition 2. *The* redundancy *of an equivalence class (a structure) is its cardinality, given by*

$$r(s) = |s| = |\{g \in \mathcal{G} \mid \Gamma(g) = s\}| \ .$$

The average redundancy *of an encoding is defined as*

$$\bar{r} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} r(s) = \frac{|\mathcal{G}|}{|\mathcal{S}|} \ .$$

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* [6]:

Definition 3. *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.
The fraction of frequent structures $h_{\mathcal{S}}$ *and the* fraction

[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.

of frequent genotypes $h_{\mathcal{G}}$ *can be defined as:*

$$h_{\mathcal{S}} = \frac{|\{s \in \mathcal{S} \mid r(s) > \bar{r}\}|}{|\mathcal{S}|} \quad \textit{and}$$
$$h_{\mathcal{G}} = \frac{|\{g \in \mathcal{G} \mid r(\Gamma(g)) > \bar{r}\}|}{|\mathcal{G}|} .$$

3.2 Analytical Analysis

Consider a d-l-1 (feed-forward) NN with d input nodes, l hidden nodes in one hidden layer, i.e., the hidden nodes are only connected to the inputs and the output. 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. Shortcuts are not relevant for the considerations[2]. In a direct encoding, there are $d \cdot l$ bits specifying the connections from the hidden neurons to the inputs. Thus, there are $2^{d \cdot l}$ 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 bit string 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:

$$|\mathcal{S}| = \begin{pmatrix} T+l-1 \\ l \end{pmatrix} . \tag{5}$$

If we additionally allow the hidden neurons to have no connection to the outputs, Eq. (5) can also be used to calculate the number of not equivalent structures for d-l-m NNs by setting $T = 2^{d+m}$. The number of different matrices accordingly rises to $2^{(d+m)\cdot l}$.

The average number of matrices per NN class is $\bar{r} = 67.6326$ for a 4-5-1 NN. Figure 3 shows the fraction of frequent structures $h_{\mathcal{S}}$ and the fraction of frequent genotypes $h_{\mathcal{G}}$ of a 4-l-1 network for $l = 1, \ldots, 30$.

For an increasing number l of hidden nodes the following observation is made: The fraction of frequent structures decreases, whereas the fraction of matrices that code for a frequent structure increases. That is the majority of genotypes (matrices) codes for a minority of structures (equivalent NNs).

Similar characteristics have been found in the mapping from RNA sequences to RNA secondary structures [12]. Nonetheless, these characteristics are not a mere consequence of the definition of rare and frequent structures. We could observe different scenarios as well. In the following, we characterize three important cases:

1. There are very few frequent structures and a tremendous number of rare ones. For example, let there be only a single frequent structure $s_0 \in \mathcal{S}$ with $r(s_0) = r_0$; the other structures are rare with $r(s) = 1$ for all $s \in \mathcal{S} \setminus s_0$. In this case the fraction of frequent structures is given by $h_{\mathcal{S}} = 1/|\mathcal{S}|$ and the fraction of frequent genotypes is $h_{\mathcal{G}} = r_0/(|\mathcal{S}| + r_0 - 1)$.
   (a) If $|\mathcal{S}|$ is much larger than r_0, both quantities, $h_{\mathcal{S}}$ and $h_{\mathcal{G}}$, go to zero. In this case, genotypes coding for rare structures dominate the search space. So there is no a priori difficulty to find a rare structure.
   (b) If r_0 is much larger than $|\mathcal{S}|$, then $h_{\mathcal{G}}$ is close to one. This leads the effect observed in Fig. 3 for larger numbers of hidden neurons. The search space is dominated by only a few frequent structures, i.e., there is a strong bias towards a small number of structures.
2. There are a lot of frequent structures and just a few rare ones; e.g., assume a single rare structure $s_0 \in \mathcal{S}$ with $r(s_0) = 1$ and $r(s) = r_s$ for all other $s \in \mathcal{S} \setminus s_0$. This leads to $h_{\mathcal{S}} = (|\mathcal{S}| - 1)/|\mathcal{S}|$ and $h_{\mathcal{G}} = ((|\mathcal{S}| - 1) \cdot r_s)/((|\mathcal{S}| - 1) \cdot r_s + 1)$, stating that both quantities approach 1 with growing $|\mathcal{S}|$. Although this is not problematic for the frequent structures it means that it is difficult to find any of the minority of rare ones.

Thus, it crucially depends on the number of equivalence classes contributing to the rare or frequent structures. If nearly all structures are either rare or frequent we get a

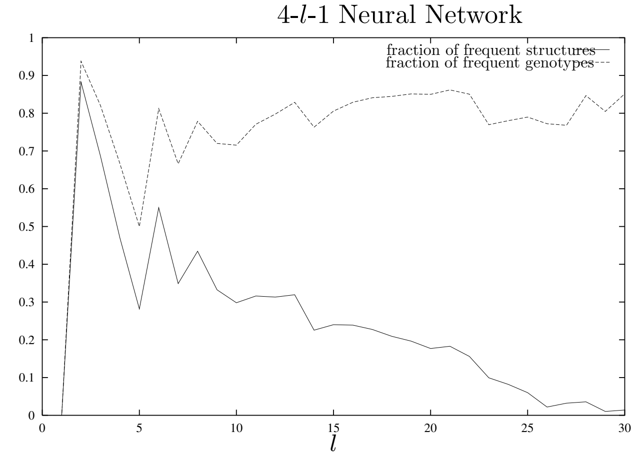

Figure 3: Fraction of *frequent* structures (solid line) and of matrices coding for these structures (dashed line) against the number of the hidden neurons.

[2] To each network one can add a shortcut connection from any input to any output node. As input and output nodes are uniquely defined by the task there is no different shortcut connection which will produce an isomorphic network. Additionally, the network with the shortcut connection and the one without will have the same number of isomorphic network topologies, as the additional shortcut does not interfere with the rest of the network structure.

situation like the one outlined in case 1 or 2. In Fig. 4 we see a histogram showing the number of structures with a given redundancy for the above analyzed 4-l-1 NN with $l = 10$. The figure reveals that there are a lot of different redundancies present.

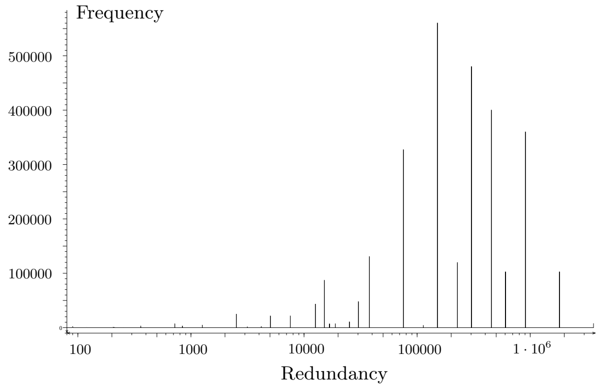

Figure 4: Histogram showing the number of equivalence classes for a 4-10-1 NN topology in dependence of their redundancy.

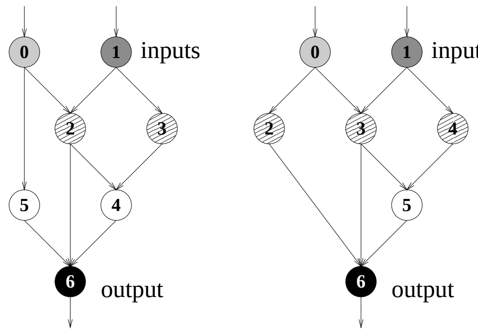

Figure 5: Two networks that can be represented as triangular matrices. In the left network neuron 5 is not ordered according to the first possible layer it can be placed in. In the right network all neurons are grouped according to the first possible layer they can be assigned to.

3.3 Random Walk Characteristics

For general (less restricted) network topologies, an analytical treatment is not tractable, so we estimate the properties of the search space by random sampling.

3.3.1 Method

We analyze search spaces of feed-forward NNs with different topology restrictions. We always consider networks with four inputs and a single output. In each random walk, the number l of hidden neurons is fixed. Different experiments are performed for various l. The search spaces are as follows:

1. NNs with a single hidden layer are an important special case and are often used as standard NN architectures. Their structure is sufficient for approximating any "well behaving" function [1]. Evolutionary structure optimization is often restricted to this basic class of NNs, although there are problems, like the two spirals problem, that are solved easier with a NN having more than one hidden layer. On the other hand, in a search space where the number of layers is not restricted, there is a strong bias towards deep network structures [14]. Therefore, we restrict the search space and allow only network architectures with a maximum number of layers. To do so we use a direct encoding but additionally specify a layer for each neuron. A neuron may only have connections to neurons in subsequent layers. Accordingly, operators are introduced that change the number of layers, namely merging of two layers and splitting of a layer. All operators regard the restrictions including the maximum number of layers. In this way topologies with few hidden layers are favored.

2. In the second set of experiments, there is no restriction on the number of layers. The networks are coded as lower triangular connection matrices. We investigate the following additional refinements of this representation:

   (a) The search space consists of all lower connection matrices coding feasible networks (each hidden neuron lies on a path from an input to an output neuron). We count all isomorphic graphs that are described by lower triangular matrices. This implies that the neurons are not uniquely assigned to layers (the layer of a neuron v can be uniquely defined by the maximum length of a path from an input neuron to v, this definition assigns the *first possible layer* to the neuron), as done in experimental setup 2b. Thus, both networks shown in Fig. 5 are included in this encoding scheme. (Finding the size of the equivalence classes consistent with this encoding is done using an enumeration scheme.)

   (b) In the next coding scheme the search space is a subset of the lower triangular matrices, because we group neurons uniquely to layers. The way we do this is to assign a neuron to the first possible layer. In this way we exclude NNs like the left one in Fig. 5. Mutations that result in NNs violating this constraint are dismissed. In this scheme it is very easy to find the cardinality of an equivalence class: Neurons that form a layer are grouped together into a color class. Thus, only neurons in the same layer can be permuted, and the size of the equivalence class is given by Eq. (1). This restriction ensures that only structures that correspond to lower triangular connection matrices are considered.

(c) Finally we do not group neurons according to layers and additionally count permutations of the sort shown in Fig. 6. They also represent correct feed-forward networks but their connection matrices are not necessarily triangular. In this case one has to keep in mind that the search space is the space of triangular connection matrices resulting in correct networks but the NNs counted as isomorphic are possibly not coded by a triangular matrix. Nevertheless, we believe that the statistics we get in our case are similar to that resulting of a complete connection matrix as search space.

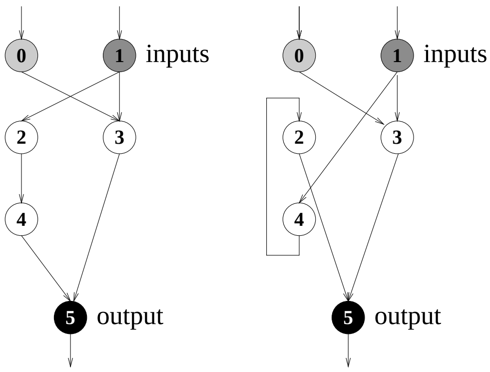

Figure 6: Two isomorphic network structures. The right one is a permutation of the left structure by changing the position of neuron 2 and neuron 4. The connection matrix of the right topology is not a triangular matrix.

For each search space, we gather 500000 genotypes by performing a random walk starting from an initial population of feasible neural networks by applying a canonical bit-flip-operator. Only genotypes representing feasible networks are accepted.

In order to visualize our results, we came up with the following notation. The structures can be ordered according to their redundancy. We want to analyze the fraction of genotypes that map to the “first” x % of the structures ordered by their redundancy, where the structures with the smallest redundancy come first. This can be formalized as follows. For each observed redundancy $i \in \{r(s) \mid s \in \mathcal{S}\}$, we can calculate the fraction of structures which have a redundancy less than or equal to i:

$$x(i) = \frac{1}{|\mathcal{S}|} |\{s \in \mathcal{S} \mid r(s) \leq i\}| \tag{6}$$

and the fraction of genotypes that are mapped to structures with redundancy less than or equal to i:

$$y(i) = \frac{1}{|\mathcal{G}|} |\{g \in \mathcal{G} \mid r(\Gamma(g)) \leq i\}| \ . \tag{7}$$

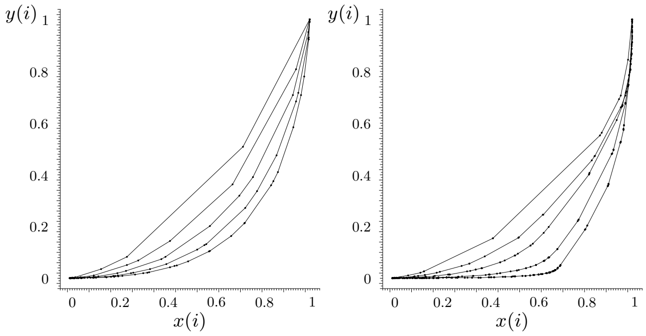

Figure 7: Fraction of genotypes against fraction of structures ordered by redundancy (see text) for 4-l-1 NNs. Left plot: one hidden layer; Right plot: the maximum number of layers is two. From top left to bottom right: l = 5, 7, 9, 11, 13.

A plot of $y(i)$ against $x(i)$ can reveal a bias of the encoding induced by isomorphic topologies. If the plot $y(i)$ vs. $x(i)$ differs from the bisecting line of the angle, i.e., $y(i) \not\approx x(i)$, then the divergence indicates a bias, because the first $y(i)$% of structures are represented by only $x(i)$% of the genotypes.

The values of $x(i)$ and $y(i)$ can be estimated by replacing $\mathcal{G}$ in Eq. (7) with the set of genotypes visited on a random walk and $\mathcal{S}$ in Eq. (6) by the corresponding set of structures.

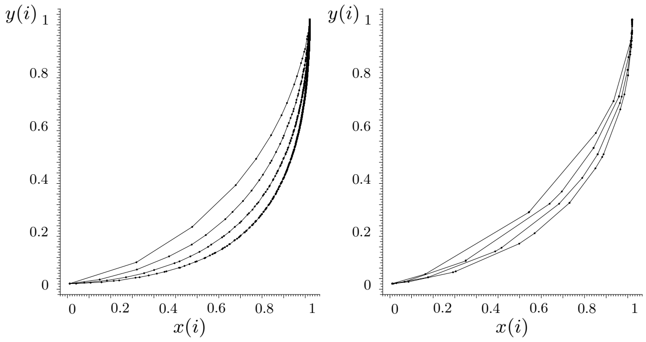

Figure 8: Fraction of genotypes against fraction of structures ordered by redundancy (see text) for a feed-forward NN with four input nodes and one output node. There are no layer restrictions. Left plot: complete triangular matrix; Right plot: Neurons assigned to first possible layer. From top left to bottom right: l = 5, 7, 9, 11.

3.3.2 Results

Results are given seperately for the discussed different encoding schemes 1, 2a, 2b and 2c.

In the first case the number of layers is restricted. If it is restricted to 1 hidden layer we get an empirical validation of the analytical results from Sec. 3.2 (with the exception that in the analytical results networks are included where a hidden node is not connected to an input.) This is shown in Fig. 7 on the left hand side with a plot of $x(i)$ against $y(i)$ for a 4-l-1 NN with one hidden layer and l = 5, 7, 9, 11, 13. (The discrete points are connected by lines for convenience). This means, a point $(0.3, 0.1)$ indicates that the first 30% of structures (ordered according to redundancies) are represented by 10% of the genotypes. For a restriction of up to 2 hidden layers the according results are shown in the right plot in Fig. 7. If the structures are restricted to networks with a single layer, we observe a bias towards redundant structures. With increasing search space, i.e., increasing number of hidden neurons, this effect gets more prominent, see Fig. 7. This is in accordance with the analytical results presented in paragraph 3.2. The same systematic behaviour is observed for 2 hidden layers.

For the other search spaces, coding via a triangular matrix, the number of layers is not restricted. For these cases we have analyzed 4-l-1 NN with $l = 4, 7, 9, 11(, 13, \dots)$.

The characteristics of the two cases where either the complete triangular matrix is used (2a) or the neurons are assigned to the first possible layer they may belong to (2b) is quite similar. In Fig. 8 plots of $x(i)$ against $y(i)$ for a 4-l-1 NN are shown. The left figure corresponds to case (2a), the right one to (2b). This reveals that the coding as a triangular matrix leads to the same problem as the layer orientated direct encoding.

On the other hand, if we do not restrict the isomorphic structures to triangular matrices we get a quite different result. Plots showing $x(i)$ versus $y(i)$ nearly give a straigth line. We also performed runs over 500000 genotypes for 4-l-1 NN with $l = 5, 7, 9, 11, 13$. For growing l the lines come increasingly closer to the bisecting line of the angle. A histogram of the redundancies and their frequencies gives the reason for this findings, see Fig. 9. We see that the vast majority of the 500000 networks

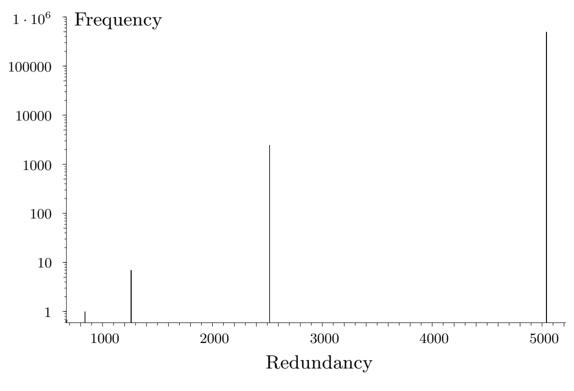

Figure 9: Histogram showing the number of networks for each redundancy found in a random walk for a 4-7-1 NN. Note the log scale on the y axis.

(exactly 497537) are represented with the same high redundancy (here 5040). There are only very few structures that have been found with a small redundancy. That means nearly all structures found are represented equally often, except for a small number of structures that are rather rarely represented. This corresponds to the considerations in Sec. 3.2, namely to point 2.

4 Applications

In this section, we discuss some possible applications of the proposed methods.

4.1 Graph Database

During an evolutionary process each generated network can be canonically labeled and the resulting canonically labeled graph can be stored in a database. Before the fitness of a new individual is determined—a computational expensive procedure in most implementations of evolutionary topology optimization—it can be tested whether this network or an isomorphic one was evaluated before by looking up the canonically labeled graph in the database. In that way, a database can be used to reduce the number of fitness evaluations. This idea was first mentioned in the work of Christoph M. Friedrich [3, 4]. The percentage of fitness evaluations that can be saved depends on the used EA and especially on the search space, i.e., whether there are topology restrictions like a maximum number of layers, cf. [14].

4.2 Competing Conventions Problem

One point that has been addressed by several researchers in the field of structure optimization of NNs is the *competing conventions* problem, also called permutation problem [2, 11, 7, 16]. 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 evolutionary optimization, especially on recombination: "Crossover among parents utilizing the same convention are likely to be successful while crossovers between dissimilar conventions are unlikely to be effective" [11]. It "makes the evolution less efficient, but also makes recombination operators more difficult to produce highly fit children" [15]. However, if only canonically labeled networks are stored in the population, there is exactly one representation of each network and the competing conventions problem taken literally can be regarded as solved. 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. The isomorphism-free representation ensures a single "convention", but does not necessarily lead to something like a "homologous" crossover between NNs.

Experiments show that there are not so many isomorphic nets present in one generation as might be expected. The number of isomorphic topologies depends on the population size, selection mechanism, operators used, and whether there are restriction on the network topologies, see Sec. 3 and [14]. So the probability of selecting two competing conventions of one net for recombination is rather small. Furthermore, we agree with many researchers in the field of evolutionary NN optimization that canonical crossover on the genotype level using a canonical representation is not an important operator for the evolution of NNs.

4.3 Bias-free Operators

In order to remove the bias stemming from isomorphic network topologies, the probability $p(g)$ that a variation operator generates an offspring $g \in \mathcal{G}$ should consider the redundancy of g, e.g.,

$$p(g) \propto \frac{1}{r(\Gamma(g))} \quad .$$

Such an operator could increase the performance of the search process, because there may be well performing rare structures that are not likely to be detected by an operator which does not take the redundancies into account and therefore tends to explore mostly frequent structures.

5 Conclusion

Calculating the canonical labeling map and the size of the corresponding equivalence class of isomorphic structures is proposed as a tool for analyzing the search space of graph representations. It does not only provide insight into theoretical properties of the search space, but also has practical applications, e.g., bias-free operators which are not affected by isomorphic structures.

However, in the case of NN structure optimization, whether a search process is influenced by the unequal representation strongly depends on the actual search space. If the search space is so large that the sample obtained during an evolutionary search does not reflect its properties or the number of different structures dominates the differences between the redundancies, then the effect of isomorphisms can be neglected.

In cases where the effect of isomorphisms matters, we propose to construct graph representations that are free of isomorphic topologies and preserve neighborhood relations with respect to "canonical" mutations in order to circumvent this problem of isomorphic topologies. This can be achieved by applying a canonical labeling map to each generated graph, e.g., after a mutation.

If good solutions are rare it might be very difficult to find them if a redundant representation is used. Thus, it seems advisable to analyze the search space and possibly get rid of the redundancies if their effects are not clear. It is difficult to know in advance what the effects of a redundant encoding will be. Regarding the RNA sequence–structure mapping a positive effect is discussed, insofar as the existing neu*t*ral networks may help to avoid local optima. Neu*t*ral networks are sets of selectively neutral neighboring genotypes, i.e., any two genotypes of a network are connected via a path of single step mutations inside the network. This means that isomorphic NNs do not constitute neu*t*ral networks if they are canonically encoded and canonical variation operators are used (bit-flip-operators, addition or deletion of a node or connection). It has to be kept in mind that a biological representation itself has been subject to the evolutionary process and therefore it is reasonable to assume that there are positive aspects of the redundancies in the biological system. In artificial structure optimization, the redundancies in most representations used exist by chance. In these cases, we believe it is sensible to remove the redundancies first and then perhaps add redundancies in a constructive manner.

We like to stress that our considerations are not restricted to topology optimization of NNs, but may also apply to a broad class of structure optimization problems.

Acknowledgments

C. Igel acknowledges support from the BMBF under grant LEONET 01IB802C4.

Bibliography

[1] J. Attali and G. Pagès. Approximations of functions by a multilayer perceptron: a new approach. *Neural Networks*, 10(6):1069–1081, 1997.

[2] R. K. Belew, J. McInerney, and N. N. Schraudolph. Evolving networks: Using genetic algorithm with connectionist learning. Technical Report CS90-174 (revised), Computer Science & Engg. Dept., Univ. of California at San Diego, USA, 1991.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] P. J. B. Hancock. Genetic algorithms and permutation problems: A comparison of recombination operators for neural net structure specification. In L. D. Whitley and J. D. Schaffer, editors, *COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pages 108–122, Baltimore, 1992. IEEE Press.

[8] B. D. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30(30):47–87, 1981.

[9] B. D. McKay. *nauty* user's guide (version 1.5). Technical Report TR-CS-90-02, Australian National University, Computer Science Department, 1990.

[10] 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.

[11] 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.

[12] P. Schuster and W. Fontana. Continuity in evolution: On the nature of transitions. *Science*, 1998.

[13] B. Sendhoff. *Evolution of Structures.* PhD thesis, Institut für Neuroinformatik, Ruhr–Universität Bochum, 1998.

[14] P. Stagge and C. Igel. Structure optimization and isomorphisms. In L. Kallel, B. Naudts, and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computing*, Natural Computing. Springer-Verlag, 2000.

[15] X. Yao. Evolving artificial neural networks. *Proc. of the IEEE*, 87(9):1423–1447, 1999.

[16] X. Yao and Y. Liu. Evolving artificial neural networks for medical applications. In *Proc. of the Australian-Korea Joint Workshop on Evolutionary Computation*, pages 1–16, Taejon, Korea, 1995.