LETTER Communicated by Charles Audet

Generation of Optimal Artificial Neural Networks Using a Pattern Search Algorithm: Application to Approximation of Chemical Systems

Matthias Ihme
mihme@stanford.edu
Department of Mechanical Engineering, Stanford University, Stanford, CA 94305, U.S.A.

Alison L. Marsden
amarsden@stanford.edu
Pediatrics Department, Stanford University, Stanford, CA 94305, U.S.A.

Heinz Pitsch
h.pitsch@stanford.edu
Department of Mechanical Engineering, Stanford University, Stanford, CA 94305, U.S.A.

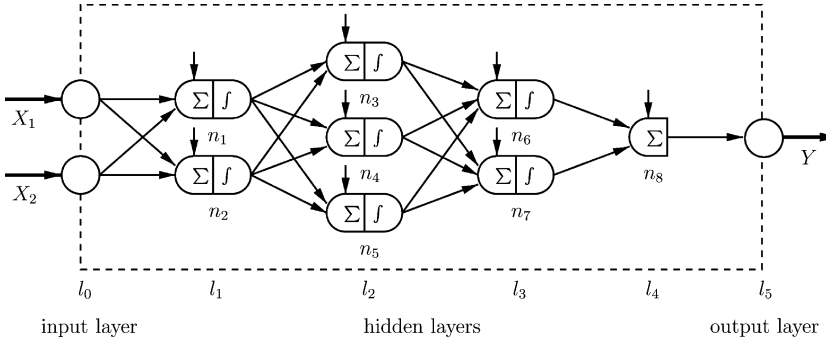
A pattern search optimization method is applied to the generation of optimal artificial neural networks (ANNs). Optimization is performed using a mixed variable extension to the generalized pattern search method. This method offers the advantage that categorical variables, such as neural transfer functions and nodal connectivities, can be used as parameters in optimization. When used together with a surrogate, the resulting algorithm is highly efficient for expensive objective functions. Results demonstrate the effectiveness of this method in optimizing an ANN for the number of neurons, the type of transfer function, and the connectivity among neurons. The optimization method is applied to a chemistry approximation of practical relevance. In this application, temperature and a chemical source term are approximated as functions of two independent parameters using optimal ANNs. Comparison of the performance of optimal ANNs with conventional tabulation methods demonstrates equivalent accuracy by considerable savings in memory storage. The architecture of the optimal ANN for the approximation of the chemical source term consists of a fully connected feedforward network having four nonlinear hidden layers and 117 synaptic weights. An equivalent representation of the chemical source term using tabulation techniques would require a 500×500 grid point discretization of the parameter space.

1 Introduction

An artificial neural network (ANN) is a computational model for storing and retrieving acquired knowledge. ANNs consist of dense interconnected computing units that are simple models for complex neurons in biological systems. The knowledge is acquired during a learning process and is stored in the synaptic weights of the internodal connections. The main advantage of neural networks is their ability to represent complex input-output relationships. They are well suited for use in data classification, function approximation, and signal processing. A comprehensive compendium of neural network theory and applications of ANNs is written by Haykin (1994).

The performance, or fitness, of an ANN is often measured according to an error between target and actual output, training time, complexity of the ANN, or in terms of other properties important for the user. Although the elemental building blocks of a neural network—neurons, nodal connections, and the transfer functions of nodes—are relatively simple, the various combinations can result in different topologies with similar or vastly different fitness characteristics. Therefore, the a priori design of a neural network with near-optimal fitness is not a trivial task and is usually guided by heuristics or trial and error. The architecture or topological structure of an ANN can be characterized by the arrangement of the layers and neurons, the nodal connectivity, and the nodal transfer functions. In this work, the class of multilayer perceptrons (MLPs) is considered, which consists of an input layer with N_I input channels, N_L hidden layers, and an output layer with N_O output channels. The number of neurons in each hidden layer is denoted by $\underline{N}_\mathrm{N}$. Nodes in the first $N_\mathrm{L} - 1$ hidden layers are characterized by a nonlinear behavior, and the nodes in the last hidden layer are linear (Haykin, 1994).

A multilayer perceptron is shown in Figure 1a. Information in this 2-3-2-1 network propagates unidirectionally from the input to the output channel. Each neuron includes a threshold that is indicated by a vertical arrow in Figure 1a. The significance of the threshold is explained in section 2. The connectivity matrix $\underline{\underline{C}}$ can be represented by the underlying directed graph and is shown in Figure 1b. The element C_{ij} in this binary matrix is unity if node j is connected to node i and zero otherwise. The first N_I rows and the last N_O rows in this matrix represent the connectivity of the input and output layers. The element C_{i0} corresponds to the threshold of neuron n_i. A similar real-valued matrix can be used for the representation of the synaptic weights in the ANN. In the case of a symmetric network, in which all neurons in a particular layer are connected to the same nodes, the nodal connectivity matrix can be contracted to a layer connectivity matrix $\underline{\underline{L}}$. The matrix entry L_{ij} indicates the connectivity between layer j and layer i. The corresponding layer connectivity matrix is shown in Figure 1c.



(a) network architecture

$$\underline{\underline{C}} = \begin{pmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&1&1&0&0&0&0&0&0&0&0&0&0&0 \\ 1&1&1&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 1&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 1&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&1&1&1&0&0&0&0&0&0 \\ 1&0&0&0&0&1&1&1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&1&0&0 \end{pmatrix}$$

(b) nodal connectivity matrix

$$\underline{\underline{L}} = \begin{pmatrix} 0&0&0&0&0&0 \\ 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \end{pmatrix}$$

(c) layer connectivity matrix

Figure 1: Multilayer perceptron. (a) Architecture of the MLP consisting of one input layer with two input channels X_1 and X_2, four hidden layers with, respectively, two, three, two, and one neurons in each layer. The neurons in the hidden layers are denoted by n_i, with $i = 1, \ldots, 8$. (b) Corresponding neural connectivity matrix. (c) Layer connectivity matrix of the MLP.

A particular topology of an ANN, to be denoted by $\mathcal{A}$, can formally be written as

$$\mathcal{A} = \mathcal{A}(N_{\mathrm{L}}, \underline{N}_N, \underline{\underline{C}}, \underline{\psi} | N_{\mathrm{I}}, N_{\mathrm{O}}, Y, J), \tag{1.1}$$

in which $\underline{\psi}$ is the vector of transfer functions for all nodes, Y is the approximated function, and J is a cost function characterizing the performance of the network. The last four arguments in equation 1.1 are constrained by the problem and the desired performance of the network. The design parameters N_{L}, $\underline{N}_{\mathrm{N}}$, $\underline{\underline{C}}$, and $\underline{\psi}$ determine the topology of the neural network

and have to be chosen in such a way that the ANN has an optimal performance characteristic. While small networks with only few connections and synaptic weights are often limited in their capability to perform a certain task, large ANNs with many nonlinear nodes can result in poor generalizability, long training time, and computationally expensive knowledge retrieval (Yao, 1999). The design of a network suitable for a given task can be formulated as an optimization problem. The choice of the method for the optimization problem is determined by the inherent properties of the ANN (Miller, Todd, & Hegde, 1989):

1. The dimension of the architecture space is infinite since the number of layers and nodes is unbounded.
2. Assessing the fitness of a network design usually requires the prior training of the network, which in itself is an optimization task and usually time-consuming.
3. The design parameters determining the topology of the network can be of both continuous and categorical type.
4. The fitness or objective function with respect to the design variables is noisy and nondifferentiable.
5. Different ANN topologies can lead to similar performance, implying that the solution space is multimodal.

In recent years, considerable research has been conducted on the evolution of topological structures of networks using evolutionary algorithms (Koza & Rice, 1991; Bornholdt & Graudenz, 1992; Tang, Chan, Man, & Kwong, 1995; Angeline, Saunders, & Pollack, 1994; Miller et al., 1989; Husken, Jin, & Sendhoff, 2005 and references in Yao, 1999). However, very little work has been carried out on evolving node-specific transfer functions or the simultaneous evolution of both nodal arrangement and transfer functions (Liu & Yao, 1996; Hwang, Choi, & Park, 1997). For the automatic design of near-optimal architectures, construction and destruction algorithms have been used (Frean, 1990; Mozer & Smolensky, 1989). This method, however, searches only in a restricted subset of possible network architectures (Angeline et al., 1994). Evolutionary algorithms (EAs) are global search algorithms and have been widely used over recent years as a method for finding optimal topological structures and node transfer functions. EAs can easily be implemented in an existing code and do not require gradient information. Despite their popularity, EAs are known to be expensive and usually lack formal convergence theory.

In this work, the generalized pattern search (GPS) method (Torczon, 1997) is used as the optimization method for the automatic design of ANNs. The GPS method is complemented by a surrogate management framework (SMF), developed by Serafini (1998) and Booker et al. (1999), in order to increase the efficiency of pattern search methods for computationally

expensive problems. In this method, the expensive cost function is approximated by a surrogate function based on Kriging (Koehler & Owen, 1996). The GPS algorithm is a derivative-free method and provides robust convergence properties (Audet & Dennis, 2003). This method has been used previously by Booker et al. (1999) for rotor-blade optimization and by Marsden, Wang, Dennis, & Moin (2004) for trailing-edge airfoil optimization. Recently Audet and Dennis (2000) extended the GPS method to problems with mixed design variables with bound constraints. This method, the so-called generalized mixed variable pattern search (GMVPS), was employed in the work of Kokkolaras, Audet, and Dennis (2000) to design a thermal insulation system. A similar method was applied for the optimal design of magnetic resonance device by Lucidi, Piccialli, & Sciandrone (2005). Here, GMVPS will be used to optimize the nodal connectivities and transfer function types of each neuron.

The main objective of this work is the development of a method that can be used to generate optimal artificial neural networks to approximate nonlinear functions that are, for instance, encountered in representing chemically reactive systems. Chemical kinetic mechanisms often comprise thousands of chemical reactions among hundreds of species. In numerical simulations of combustion systems, for example, the direct solution of transport equations for all these species is usually not feasible, partly because of the large range of chemical timescales. Alternative approaches, such as intrinsic low-dimensional manifolds (Maas & Pope, 1992) or computational singular perturbation (Lam & Goussis, 1988), have been developed, in which a part of the chemical species are projected onto lower-dimensional manifolds, resulting in a reduced chemical reaction mechanism. However, the necessary introduction of certain assumptions, required for the mechanism reduction, can result in a degradation of the accuracy and generality.

Tabulation techniques, on the other side, are often employed when the number of independent scalars is small. This is due to the fact that the memory requirement and data retrieval time increase with the number of independent parameters, imposing drastic restrictions if more than three independent parameters are used. Tabulation is often employed in the application of the steady flamelet model (Peters, 1984), in which all thermochemical quantities are represented using only two scalars. However, in the case of turbulent combustion and when radiation or pollutant formation is of importance, accurate predictions might require the number of independent scalars to grow to five or even more (Peters, 2000).

In situ adaptive tabulation (ISAT) (Pope, 1997) and solution mapping using piecewise polynomial approximation (PRISM) (Tonse, Moriarty, Brown, & Frencklach, 1999) are other examples of tabulation methods. Contrary to the conventional tabulation, in which the complete accessible thermochemical phase space is parameterized, in ISAT and PRISM only the accessed state is computed and included in the table for future reuse.

Over recent years, ANNs have successfully been employed for the approximation of chemical systems (Christo, Masri, & Nebot, 1996; Christo, Masri, Nebot, & Pope, 1996; Blasco, Fueyo, Dopazo, & Ballester, 1999; Blasco, Fueyo, Dopazo, & Chen, 2000; Blasco, Fueyo, Larroya, Dopazo, & Chen, 1999; Chen, Blasco, Fueyo, & Dopazo, 2000; Flemming, Sadiki, & Janicka, 2005). Important advantages of ANNs over tabulation methods are its modest memory requirement and cost-effective function representation. However, a major drawback of ANN application to chemistry approximation is the limited control of approximation error. To overcome this shortcoming, a methodology for the generation of optimal ANNs with local error control is presented in this letter.

Motivated by the chemistry application, test examples are performed using functions that closely resemble the chemical characteristics of combustion systems. Nevertheless, the proposed method for the optimization of the network architecture is general in nature and applicable to other problems of interest.

The remainder of the letter is organized as follows. Section 2 discusses the ANN model and describes the training process. The GPS and GMVPS methods are presented in section 3. These algorithms are then applied in the optimization of neural networks. In the first step, only the numbers of neurons per layer are optimized, and the connectivity matrix and type of transfer function are fixed. This optimization problem is solved using the surrogate management framework with standard GPS. In the second step, the nodal transfer function and connectivity matrix are introduced as free optimization parameters along with the numbers of neurons. Results for different optimal networks are presented. In the last step, an example of the approximation of a reactive system is presented, in which the nodal arrangement, transfer function, and connectivity are parameters and optimal networks are generated using the GMVPS method. The performance of the automatically generated network is compared with results obtained using a conventional tabulation technique. The letter concludes with a discussion.

2 Artificial Neural Network

The general structure of an MLP is shown in Figure 1a. This particular network consists of $N_\mathrm{I} = 2$ input channels and $N_\mathrm{O} = 1$ output channels and $\mathcal{A} : \mathbb{R}^{N_\mathrm{I}} \rightarrow \mathbb{R}^{N_\mathrm{O}}$. Note that the input nodes do not represent neurons in the conventional sense. Here they merely perform the task of normalizing the input data so that the input signal feed into the network is in the interval $[-1, 1]^{N_\mathrm{I}}$. The output y_i of each neuron i in the hidden layers is computed according to

$$y_i = \psi_i \left(\sum_{j=0}^{N_\mathrm{C}} C_{ij} \omega_{ij} x_{ij} \right), \tag{2.1}$$

where x_{ij} is the input signal and N_C denotes the number of connections in the network. The synaptic weights are denoted by ω_{ij}. Each neuron includes a threshold ω_{i0} that is connected to a constant input signal $x_{i0} = -1$. The effect of the threshold is to lower the net input on the transfer function (Haykin, 1994). A sigmoidal function

$$\psi(s) = a_1 \tanh(b_1 s), \tag{T1}$$

a cyclometric function

$$\psi(s) = a_2 \operatorname{atan}(b_2 s), \tag{T2}$$

a logarithmic function

$$\psi(s) = \begin{cases} \log(1+s) & \text{if } s \geq 0, \\ -\log(1-s) & \text{if } s < 0, \end{cases} \tag{T3}$$

a signum function

$$\psi(s) = \operatorname{sgn}(s), \tag{T4}$$

or a linear function

$$\psi(s) = s \tag{T5}$$

is commonly used as a transfer function. The parameters a and b in the transfer functions T1 and T2 are adjustable by the user. Transfer functions T1 and T2 are shown in Figure 2 and will be used in section 4.

The synaptic weights in the network are adjusted during the training process, which in itself represents an optimization problem and can be written as

$$\min_{\omega \in \mathbb{R}^{N_\omega}} E(\omega), \tag{2.2}$$

where N_ω denotes the number of synaptic weights in the network. In the above problem, $E : \mathbb{R}^{N_O \times N_t} \to \mathbb{R}$ is a measure of the error between the actual and desired output of the network, which can be written as

$$E = \frac{1}{2\,N_t} \sum_{j=1}^{N_t} e^t(j), \tag{2.3}$$

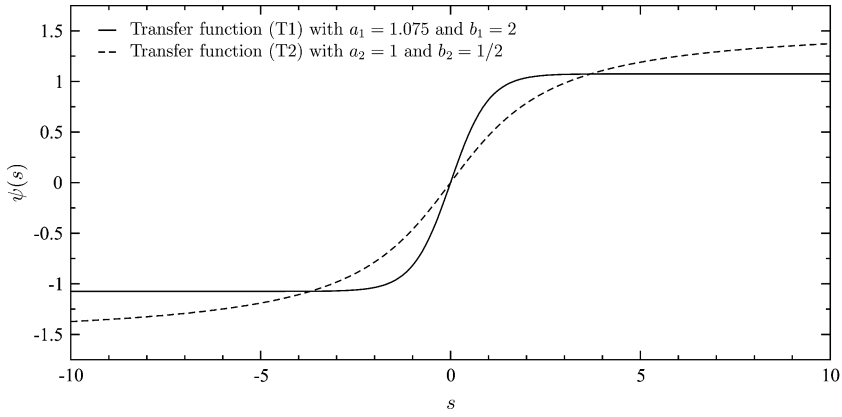


Figure 2: Transfer functions.

with

$$e^{\mathrm{t}}(j) = \sum_{i=1}^{N_{\mathrm{O}}} \left[Y_i(j) - Y_i^{\mathrm{t}}(j) \right]^2 , \tag{2.4}$$

and $Y_i^{\mathrm{t}}(j)$, a function of the synaptic weights, represents the jth training sample of the output signal i. The number of training samples is denoted by N_{t}. The learning process, in which the weights are iteratively adjusted, is guided by the knowledge of the desired input-output examples and is called supervised learning. A Levenberg-Marquardt algorithm (Hagan, Demuth, & Beale, 1996) has been used for the optimization of the synaptic weights. This algorithm is a trust region Gauss-Newton method, and its fast convergence makes it suitable for training neural networks.

The main advantage of ANNs is their generalizability, meaning their accurate performance on new data. In this respect, ANNs are different from tabulation methods in which discrete values of a particular function of interest are memorized and stored in a table. A good generalization of a particular ANN is dependent on the complexity of the function, the architecture of the network, and the size and information content of the training set. Networks with only few hidden neurons and synaptic weights are restricted in the approximation of complex functions, resulting in poor generalizability, which is characterized by so-called underfitting. A network that is too complex, meaning that there are considerably more synaptic weights than training samples, can result in possible overfitting and nonsmooth function approximation.

The appropriate size of the training set is crucial for the design of an optimal network and is usually determined by two factors. First, the

a priori knowledge of the function to be approximated allows an assessment of the complexity and shape of the function. Based on this information, the minimum size of the training set required for the description of the functional shape can be determined. A second constraint on N_t is imposed by the size of the network and the number of synaptic weights. The size of the training set for a fixed architecture with N_ω synaptic weights is given by

$$N_t \geq \alpha N_\omega \ . \tag{2.5}$$

The value of α is chosen based on experience and analysis and typically ranges from approximately 30 for noisy training data down to 2 to 5 for the approximation of analytical functions. The lower bound is chosen to avoid the possible occurrence of overfitting, which can occur if N_t is significantly smaller than αN_ω.

The generalization potential of a trained ANN is assessed using test samples. These samples are used after training to evaluate the ability of the ANN to approximate untrained samples. In this process, the performance of the ANN can be evaluated by means of the following cost function,

$$J(\mathcal{A}) = \log_{10}\left(\sqrt{\frac{1}{N_s}\sum_{j=1}^{N_s}(e^s(j))^2}\right), \tag{2.6}$$

where N_s is the number of test samples. The evaluation of the cost function requires that the synaptic weights in $\mathcal{A}$ are fully adjusted during a preceding training process. The decadic logarithm in equation 2.6 is introduced here only for convenience to enlarge the resolution of the cost function. In order to allow an objective comparison of the fitness between different network architectures, e^s is normalized to the interval $[0, 1]$.

The resulting optimization problem may be formulated as

$$\begin{array}{ll} \min_{N_L, \underline{N}_N, \underline{\underline{C}}, \underline{\psi}} & J(\mathcal{A}) \\ \text{subject to} & N_L = N_L^{\max}, \\ & N_{N,i} \in \{0, 1, \ldots, N_N^{\max}\}, \qquad i = 1, 2, \ldots, N_{L-1}, \\ & N_{N,N_L} = 1, \\ & \underline{\underline{C}} \in \mathcal{C}, \quad \underline{\psi} \in \mathcal{P}, \end{array}$$

where $\mathcal{P}$ is the finite list of transfer functions and $\mathcal{C}$ denotes the finite set of possible connectivity matrices. In the context of the GPS method, continuous variables refer to real-valued or integer-valued quantities, which are countable. In contrast, categorical variables are quantities that belong to a set of possible candidates that are not countable. In our case,

the continuous variables are N_L, and $\underline{N}_N$, and the categorical ones are $\underline{\underline{C}}$ and $\underline{\psi}$.

3 GPS Method and the GMVPS Algorithm

The GMVPS method is an extension of the GPS method (Torczon, 1997), and has been developed by Audet and Dennis (2000) to handle mixed variable problems with bound constraints. In this case, the optimization of a network architecture can be represented as a mixed variable problem in which the hidden layers and neurons are (continuous) integer-valued parameters and the transfer functions and connectivity matrix are of categorical type. Categorical variables must take on values from a predefined list or discrete set of elements. Optimization in this mixed variable space is performed to find a network architecture with optimal fitness, subject to certain constraints. In this context, it needs to be pointed out that the pattern search method does not guarantee that the solution is a global minimum. The limit point of the sequence of iterations, however, corresponds to a local optimal solution, which is defined with respect to a set of neighbors. The neighborhood is specified by the user and accounts for the variation of continuous and categorical variables. A rigorous definition of local optimality for a mixed variable problem is given by Audet and Dennis (2000).

The pattern search algorithm is a derivative-free mesh-based method. The algorithm generates a sequence of iterates, whose cost function is nonincreasing (Audet & Dennis, 2000). All points at which the cost function is evaluated are restricted to lie on a mesh. The algorithm proceeds in two stages: a search and a poll step. The search step allows a mesh-restricted local and global exploration of the parameter space. In this step, a finite number of search points are evaluated with the objective of identifying a region with a reduced cost function. For instance, random sampling of the parameter space using Latin hypercube sampling (LHS) (McKay, Beckman, & Conover, 1979) or a genetic algorithm can be employed. A considerable cost reduction of the search step can be achieved by employing a less expensive surrogate function. The surrogate function is often an approximation to the cost function. The shape of the function is continuously updated by incorporating all previously evaluated points of the cost function. This surrogate function is then used to identify a new point with a potentially lower cost function.

In this work, kriging approximation is employed as the surrogate function. Kriging is a statistical method and is based on the use of spatial correlation functions. Its multidimensional extension makes this method attractive for optimization problems with several parameters. A summary of the kriging approximation is given by Marsden et al. (2004). Details of implementation are described by Lophaven, Nielsen, & Søndergaard (2002). Note that kriging approximation as a surrogate is used here only for the optimization problem in the continuous parameter space.

In the case of an unsuccessful search step, a poll step in a discrete neighborhood of the incumbent point with the lowest cost function is executed. All continuous-valued points are restricted to lie on a mesh that is constrained to the parameter space. For a mixed variable problem, the mesh M at iteration k is defined to be the direct product of the categorical variable space Ω^c and the lattice in the continuous variable space,

$$M_k = \Omega^c \times \left\{ \underline{\eta}_k + \Delta_k \underline{\underline{D}}\, \underline{\varsigma} : \underline{\varsigma} \in \mathbb{N}^{N^D} \right\}, \tag{3.1}$$

where $\Delta_k > 0$ is the mesh size parameter and $\underline{\underline{D}}$ is an $N^c \times N^D$ matrix whose columns form a positive spanning set. If $\underline{\underline{I}}$ denotes the identity matrix and $\underline{\iota}$ is the vector of ones, then $\underline{\underline{D}}$ is typically chosen as $\underline{\underline{D}} = [\underline{\underline{I}}, -\underline{\iota}]$ or $\underline{\underline{D}} = [\underline{\underline{I}}, -\underline{\underline{I}}]$ (Audet & Dennis, 2000).

The poll step consists of a local search in the mesh neighborhood around the current best point and, in the case that the categorical variable space is not empty, also in the set of the categorical neighbors. Polling is conducted in three steps (Audet & Dennis, 2000):

- Polling with respect to the continuous variables and fixed categorical parameters
- Polling in the neighborhood of categorical variables and fixed continuous parameters
- Extended polling around the neighborhood of points whose cost function is close to the incumbent value

The three poll steps are schematically shown in Figure 3. The abscissa represents the continuous variable space, and the ordinate corresponds to the categorical space. The continuous poll step is an evaluation of adjacent mesh points forming a positive spanning set. This step is augmented with the evaluation of points in the discrete neighborhood. If these poll steps are unsuccessful, extended polling is performed. Extended polling around promising points is triggered when the previous poll steps are unsuccessful. A promising point ζ for the extended polling is defined as

$$J(\eta_k) < J(\zeta) \leq (1 + \xi) J(\eta_k), \tag{3.2}$$

where $\xi > 0$ is the poll trigger. If $J(\zeta)$ is close to $J(\eta_k)$ of the current best point, polling around the continuous neighborhood of ζ is performed with the expectation of finding a better point. If a point ζ^* is found with $J(\eta_k) < J(\zeta^*) < J(\zeta)$, extended polling around this point is continued. This continues until either a new incumbent is found or all points in the extended poll set are inferior to the incumbent. If all poll steps are unsuccessful, the mesh is refined, and a new iteration, starting with a search step, is performed. More details on the algorithm and convergence proofs can be found in Audet and Dennis (2000) and Abramson (2004).

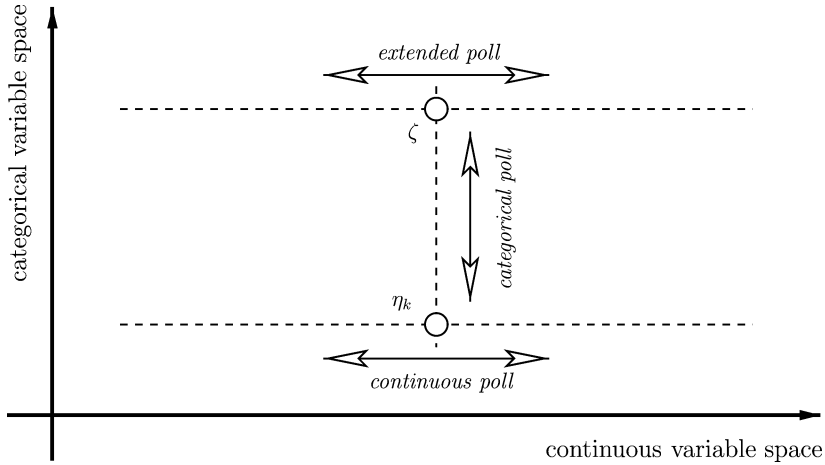


Figure 3: Three poll steps in the GMVPS algorithm. The current best point in iteration k is denoted by η_k.

4 Results

In this section, different aspects of network optimization are explored, including optimization of the nodal arrangement, type of transfer function, and connectivity. First, the nodal arrangement of the ANN is optimized using the standard GPS together with surrogate approximation. In this case, the hidden layers and neurons of the network are free parameters (see section 4.1). In the next case (see sections 4.2 and 4.3), the transfer function type and connectivity matrix are added as optimization parameters, and the network dependence on N_L, $\underline{N}_N$, and $\underline{\underline{C}}$ is optimized subject to the cost function constraint. The nodal transfer function and the connectivity are parameters of categorical type and therefore require the application of the GMVPS method. In section 4.4, the optimization method is applied to the approximation of chemical systems with optimal ANNs. Results for two different optimization processes are discussed.

To demonstrate the use of the method, a network with a scalar output, dependent on two input channels, is optimized. Mimicking a chemical reaction rate dependent on two input channels, the following analytical function is used:

$$Y = \exp\left\{-\sum_{i=1}^{N_I=2} \frac{(X_i - X_i^0)^2}{\sigma_i}\right\}, \tag{4.1}$$

with $\underline{X}^0 = (0.5, 0.5)^T$, $\underline{\sigma} = (0.01, 0.1)^T$, and $\underline{X} \in [0, 1]^2$.

4.1 Optimal Nodal Arrangement.

4.1.1 Example for One Nonlinear Hidden Layer. In this section, the GPS method is applied to optimize the number of neurons in a fully connected MLP. The number of neurons in the first hidden layer is a free parameter, and its bounds are $0 \leq N_{\mathrm{N},1} \leq 32 = N_{\mathrm{N}}^{\max}$. A hyperbolic tangent function with $a = 1.075$ and $b = 2$ is used as the transfer function. The numbers of test samples and training samples are equal and set to 350 so that the coefficient α in equation 2.5 is $\alpha \geq 2.7$. It can be expected that in the limit of small $N_{\mathrm{N},1}$, the network suffers from underfitting, and overfitting might be observable for large $N_{\mathrm{N},1}$ due to small α-values. For the training of the individual networks during the GPS optimization, at most 200 iterations in the Levenberg-Marquardt algorithm are used.

The training process and the evaluation of the network fitness are not deterministic and contain stochastic components. First, the initialization of the synaptic weights from a uniform distribution (Haykin, 1994) can result in different trajectories in the search space and usually lead to different local minima. A second random component is introduced by the choice of the training and test data, which are chosen from an infinite-dimensional space. It must be appreciated that these two stochastic components can result in different outcomes between different optimization processes. In order to reduce the stochastic contribution in the assessment of the performance of the network structure for this investigation, the L_2-norm rather than the L_∞-norm is used as cost function. The synaptic weights are initialized with a constant value of 0.25. This allows us to reproduce the results for a given set of training and test samples in this case. These restrictions, however, are not critical for practical applications.

The evolution of the optimization algorithm is shown in Figure 4. The surrogate function (solid line) is determined by the surrogate points (circles) and comprises all locations at which the cost function has previously been evaluated. Points evaluated during the search step, obtained from the minimization of the surrogate function, are denoted by square symbols. Locations that are evaluated during the poll step are denoted by diamonds, and the current best value for $N_{\mathrm{N},1}$ resulting in the best ANN performance, is indicated by a star. In the first iteration, Figure 4a, the minimum of the surrogate, evaluated during the search step, corresponds to the current best point. The search step is declared unsuccessful, and polling around the previous optimal location is performed. The poll step does not result in a reduction of the cost function. The mesh size is reduced to $\Delta = 4$ in the succeeding iteration and the surrogate is updated (see Figure 4b). Neither the search nor the poll step is successful in the second iteration, and the mesh size is further reduced to $\Delta = 2$ (see Figure 4c). The minimum of the newly updated surrogate function is computed and the cost function is evaluated, resulting in a reduction of $J(\mathcal{A})$ and therefore a better ANN structure. Both search and poll steps in the fourth iteration (see Figure 4d)

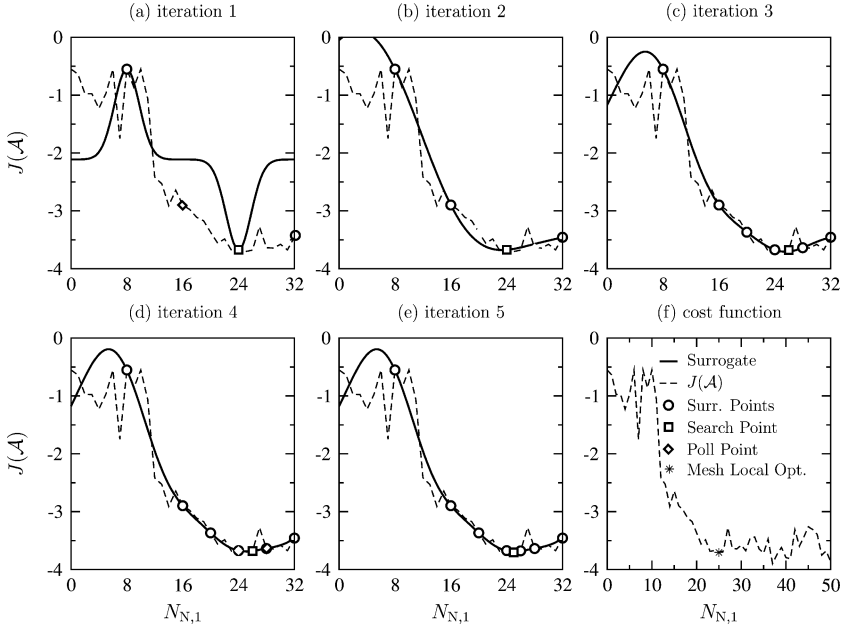


Figure 4: Evolution of the GPS optimization for one-parameter case.

are unsuccessful, and the mesh size is reduced to $\Delta = 1$ in iteration 5. From the updated surrogate function, the search step determines a new minimum of the surrogate function for $N_{N,1} = 25$. The cost function for this location is evaluated, resulting in further improvement of the network fitness. The GMVPS algorithm identifies the mesh local optimizer at $N_{N,1} = 25$.

The GPS algorithm requires only eight function evaluations for the optimization of this particular network with one nonlinear hidden layer. A function evaluation comprises both the training phase and fitness evaluation during the test process. It is also interesting to point out that two search steps (iteration 3 and 5) lead to a reduction in the cost function, and all poll steps were unsuccessful.

It can be seen in Figure 4e that the function evaluations are mainly clustered around the mesh local optimizer. This pile-up of points in the kriging function can lead to a degradation of the surrogate function (Marsden et al., 2004). This can be prevented by including space-filling points, which necessarily do not correspond to optimal points.

The evolution of the cost function for $0 \leq N_{N,1} \leq 50$ is shown in Figure 4f. For small $N_{N,1}$, the network suffers from underfitting, resulting in poor generalization. With increasing number of neurons and hence synaptic weights, $J(\mathcal{A})$ decreases and stays approximately constant for $N_{N,1} \geq 35$. A further increase of the number of synaptic weights results in only marginal

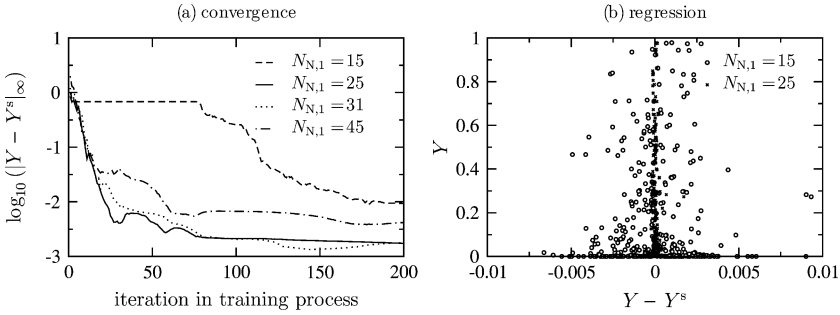


Figure 5: Convergence during the training process and regression analysis for different network structures.

improvement of the fitness. This can mainly be attributed to the limited complexity of the network structure (one hidden nonlinear layer) and possible overfitting for large $N_{N,1}$. Overfitting is demonstrated in Figure 5a, in which the L_∞-norm of e^s is evaluated after each iteration in the training process. The increase in the error norm, for instance, for the network with $N_{N,1} = 45$ neurons after 75 iterations, is an indication of network overfitting. Early stopping, as described by Haykin (1994), could be used to prevent this effect. However, early stopping can result in suboptimal networks, as shown by the convergence history of the network for the mesh local optimizer. After the increase of the error after about 30 iterations, the synaptic weights are further adjusted and the error decreases. Figure 5b shows a regression plot for $N_{N,1} = 15$ and 25, respectively. The output value $Y(j)$ is plotted as a function of $(Y(j) - Y^s(j))$ for all test samples $j = 1, \ldots, N^s$. The standard deviation of Y for the mesh local optimizer is 3.8×10^{-8}.

4.1.2 Example for Multilayer Networks. The GPS algorithm is now applied in the optimization of a network with three nonlinear hidden layers. Note that an additional fourth layer contains a single linear neuron. The bounds on the number of neurons in each layer are $\underline{0} \leq \underline{N}_N \leq \underline{16}$, which results in at most 609 adjustable synaptic weights. The training and test set each contains 1000 randomly chosen samples. The synaptic weights are initialized with 0.25. The initial mesh size is set to $\Delta = 8$, and four initial data points obtained from an LHS are used. To facilitate a more extensive search step, three points are evaluated in each search step. The first point is the nearest mesh location corresponding to the surrogate minimizer. The second search point is chosen from the neighborhood of the current best point using the surrogate function as a predictor. Note that this point belongs to the set of points evaluated during the succeeding poll step in the case that the search step fails. The third point is randomly chosen and used to improve the quality of the surrogate function.

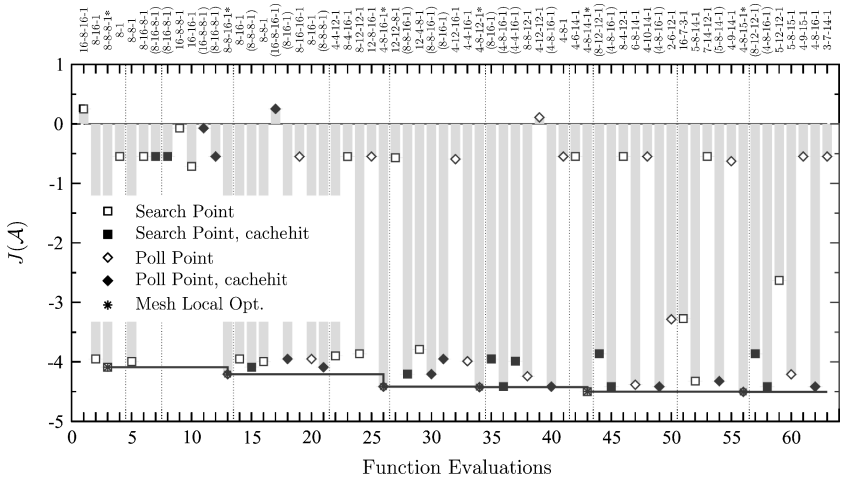


Figure 6: Evolution of the cost function for three-parameter optimization. The initial mesh size is $\Delta = 8$, and 1000 randomly chosen samples are used for training and testing. The synaptic weights are initialized with a constant value of 0.25. The particular network architecture at each function evaluation is shown at the top of the graph, and values in parentheses refer to cachehits.

The evolution of the cost function is shown in Figure 6. The number of function evaluations includes the initial search points and also points that have previously been evaluated. These points are indicated by solid symbols and are referred to as cachehits (Abramson, 2004) and do not need to be reevaluated. Square symbols correspond to search points, and poll points are marked by diamonds. The current best point is indicated by a star and also marked with an asterisk in the network architecture, shown at the top of the graph. The vertical dotted lines indicate the beginning of a new iteration.

Generally it can be observed that networks with different nodal arrangements exhibit vastly different fitness. For this particular example, it is also demonstrated that larger networks with more synaptic weights, and hence degrees of freedom, do not necessarily result in better generalizability. These observations support the need for a methodology to automatically design optimal ANNs subject to specified constraints or desired network characteristics.

Ten iterations are required to find the mesh local optimizer, which is found to be a 4-8-15-1 network. In total, 63 network architectures are used in the GPS algorithm, and 22 of them have been previously computed (cache hits). This implies that only 1% of the total 4097 ($= (N+1)^3 - 3(N^2+N)$ with $N = 16$) possible configurations are used to find the mesh local optimal structure. Note that no early stopping criterion has been used, and the GPS

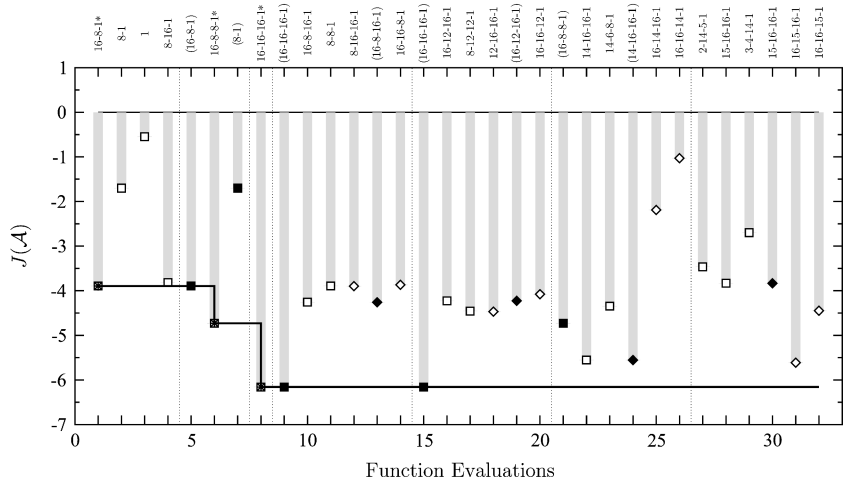


Figure 7: Evolution of cost function for three-parameter optimization and random initialization of the synaptic weights. For the legend, see Figure 6.

algorithm stops when the smallest mesh size, $\Delta = 1$, is reached and no further mesh refinement is possible. It is interesting to point out that one search step (iteration 7) and four poll steps (iterations 2, 4, 5, and 9) were successful in finding a new ANN with improved fitness.

The importance of the initialization of the synaptic weights is demonstrated in Figure 7. Compared to the previous example, where all synaptic weights are initialized with 0.25, here a random initialization is used. The set of training and test samples is identical for both realizations. In this case, the optimal ANN corresponds to a 16-16-16-1 network with a cost function of $J(\mathcal{A}) = -6.16$, resulting in an increase of the fitness by 37% compared to the previous example. The mesh local optimum is found in the second iteration. As no further point leads to a better network fitness, the algorithm terminates after 32 function evaluations.

4.2 Optimal Transfer Function. The transfer function for each neuron in the network has so far been chosen a priori and fixed during the network optimization. However, it is well known that the transfer function has a significant effect on the performance and generalizability of an ANN. This observation is also substantiated by the example shown in Table 1. For this particular network with five nonlinear hidden layers, a fixed number of eight neurons per layer is used. By keeping all other parameters identical, different transfer functions are used, and their effect on the network performance is assessed. For the example shown in Table 1, two different transfer functions are employed: the sigmoidal function (T1) with $a_1 = 1.075$ and

Table 1: Comparison of the Cost Function for an 8-8-8-8-8-1 ANN with Different Transfer Functions.

Transfer Function	$J(\mathcal{A})$
(T1)-(T1)-(T1)-(T1)-(T5)	−0.534
(T2)-(T2)-(T2)-(T2)-(T5)	−3.119
(T1)-(T2)-(T2)-(T2)-(T5)	−3.844

Notes: The parameters for the transfer function are identical to these shown in Figure 2. The synaptic weights are initialized with 0.25, and 500 samples are used for training and testing.

$b_1 = 2$, and a cyclometric function (T2) with $a_2 = 1$ and $b_2 = 1/2$. Poor network fitness is obtained when the sigmoidal function is exclusively used for all nonlinear hidden layers. The performance can be substantially increased when the transfer function is changed to a cyclometric function. A further improvement is possible when the transfer functions are arranged as shown in the last row of Table 1. In this case, the neurons in the first hidden layer use a sigmoidal transfer function, and the nodes in the other nonlinear hidden layers are of a cyclometric type.

In this section, the choice of the optimal transfer function is addressed, and examples are presented for which the types of transfer functions are considered to be free parameters. The GMVPS algorithm is employed to identify a network with optimal fitness characteristics.

In the context of the optimization problem, the nodal transfer function can be considered in two ways. In the first approach, the type of transfer function (e.g., sigmoidal transfer function, T1) is chosen a priori, and the free parameters a_1 and b_1 are adjusted so that the ANN has optimal performance. In this sense, the parameters are of a continuous nature, and the method presented in section 4.1 can be employed by a simple extension of the parameter space.

In the second approach, different types of transfer functions with fixed parameters are used during the optimization process. In this case, the transfer functions are of a categorical type and the GMVPS algorithm is employed. In the most general case, the transfer function for each neuron can be considered as a free categorical parameter and adjusted to optimize the network fitness. However, in order to reduce the complexity, the search space is considerably reduced, and all neurons in a layer are assigned to have the same transfer function. This constraint will be retained in the following examples. The method presented can be extended in a straightforward way to the more general case.

In the following example, the architecture of a fully connected feedforward network is optimized. The maximum number of hidden nonlinear layers is five, and the maximum number of neurons in each layer is restricted

Table 2: Comparison of Results for the Optimization Problem with Categorical Parameters.

Neurons	Transfer Function	$J(\mathcal{A})$	Evaluations	Iterations
7-5-7-8-1	(T2)-(T2)-(T2)-(T1)-(T5)	−4.372	125/145	7/9
4-4-8-1	(T1)-(T1)-(T1)-(T5)	−3.906	3/32	1/4
8-8-8-1	(T2)-(T2)-(T2)-(T2)-(T5)	−4.029	12/30	4/6

Notes: The parameters for the transfer function are identical to those shown in Figure 2. The synaptic weights are initialized with 0.25, and 500 samples are used for training and testing.

to eight. In this example only T1 and T2 are used as possible candidates for the transfer function. The transfer function in the last linear layer is fixed and of type T5. Contrary to the method presented in section 4.1, where a surrogate function has been used during the search step to identify a candidate for a network with optimal fitness, here two LHS points are used to find a global minimum of the cost function. A third point is a randomly chosen member from the continuous poll neighborhood.

The network with the optimal fitness is determined with respect to the choice of the categorical neighborhood. If, in the present case, all 2^5 possible categorical neighbors are incorporated in the poll step, a more global solution can be found, although at extraordinary computational costs. Here the set of categorical neighbors includes the following set of combinations:

- The transfer functions for the neurons in all layers are identical, of type T1 or T2.
- Except the neurons in one layer, the transfer function for the neurons is identical.
- All even-numbered layers contain a transfer function T1, and the odd-numbered layers are of type T2.
- All even-numbered layers contain a transfer function T2, and the odd-numbered layers are of type T1.

The poll trigger value is set to $\xi = 0.1$.

In Table 2, the result for this example is compared to the optimization problem for which the transfer functions are fixed. The network with the mesh local optimal fitness is a 7-5-7-8-1 ANN. The first three layers have a sigmoidal transfer function, and the fourth layer has a cyclometric function. This architecture is found after 125 function evaluations and seven iterations. A total of 145 different network architectures are evaluated in nine iterations.

Note that the same sets of training and test data are used for the cases shown in Tables 1 and 2. For the optimization problem in Table 1 (third row), the number of neurons is kept constant, and only the transfer functions are dynamically adjusted. The last two rows in Table 2 show the results of an

optimization where the transfer function is fixed, and only the nodal arrangement is a free parameter. Depending on the type of transfer function, two different networks with slightly different fitness are obtained. These results must be compared with the fitness characteristics for the case when both the neurons and nodal transfer function are simultaneously optimized, resulting in a further improvement of the cost function. This example indicates that the neural arrangement and the transfer functions are mutually dependent and require a joint optimization.

4.3 Optimal Connectivity. For the approximation of static functions, such as of interest in this letter, fully connected feedforward MLPs are often used. In this case, the connectivity is predetermined. Greater flexibility can be obtained when the networks includes feedbackward connections. These so-called recurrent neural networks (RNNs) are employed for approximating dynamic functions encountered in speech recognition or stock market prediction.

In this section, results are presented using the GMVPS algorithm for the optimization of network topology for which the connectivity and the number of layers and neurons are parameters. For simplicity, the nodal transfer function is predefined and of type T1. We are mainly interested in the approximation of static functions and therefore exclude RNNs from the set of possible candidates.[1] The connectivity is restricted to feedforward networks with arbitrary connectivity: with $C_{ij} = 1$ for $j < i$. A further restriction on the connectivity is imposed by preserving the symmetry of the network, requiring that all neurons in a particular layer have the same connectivity. This is certainly not necessarily the case for the first hidden layer, which usually serves to partition the input space into regions (Fausett, 1994). This constraint is mainly imposed to reduce the size of the categorical neighborhood. Under this premise, the nodal connectivity $\underline{\underline{C}}$ can be simplified and represented by the layer connectivity $\underline{\underline{L}}$, which is shown in Figure 1c and contains only information about the connectivity between all layers in the network. The set of categorical neighbors includes the following connectivities:

- Fully connected feedforward network
- Fully connected feedforward network with $L_{ij} = 1$ for $j = i - 1$ where i represents any hidden layer
- Fully connected feedforward network with $L_{ij} = 1$ for $j = i - 2$ where i represents any hidden layer

[1] This restriction is mainly imposed by the Levenberg-Marquardt algorithm, used for the network training. This algorithm cannot be employed for the optimization of RNNs. Note, however, that the optimization methodology presented here is nevertheless applicable to RNNs.

Table 3: Network Architectures for Different Conditions and Optimal Connectivity.

Network	$\underline{\psi}$	Neurons	$\underline{\underline{L}}$	$J(\mathcal{A})$	Evaluations	Iteration
4-4-4-4-4-1	(T1)	4-4-4-4-3-1	$\begin{pmatrix} 0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0\\ 1&1&0&0&0&0&0&0\\ 1&0&1&0&0&0&0&0\\ 1&0&0&1&0&0&0&0\\ 1&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0 \end{pmatrix}$	−3.889	43/63	6/7
8-8-8-8-8-1	(T1)	6-4-8-1	$\begin{pmatrix} 0&0&0&0&0&0\\ 1&0&0&0&0&0\\ 1&1&0&0&0&0\\ 0&0&1&0&0&0\\ 0&0&0&1&0&0\\ 0&0&0&0&1&0 \end{pmatrix}$	−4.189	28/56	4/6
16-16-16-1	(T1)	6-16-1	$\begin{pmatrix} 0&0&0&0&0\\ 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0 \end{pmatrix}$	−4.303	28/38	7/8

- Fully connected feedforward network with $L_{ij} = 1$ for $j = i - 3$ where i represents any hidden layer
- Fully connected feedforward network with $L_{i0} = 1$ and i represents any hidden layer
- Fully connected feedforward network with $L_{i0} = 1$ for all i hidden layers

Numerical experiments for different networks have been performed. The synaptic weights are initialized with 0.25, and 500 samples are used for training and testing. The results of the optimization process are summarized in Table 3. The first two columns in Table 3 denote the constraint on the network: the maximum number of neurons per layer and the type of transfer function. The numbers of neurons, connectivity, and cost function for the optimized network are shown in the table. Note that a sigmoidal transfer function is used here. The numbers of function evaluations and iterations, used to identify the mesh-local optimizer, are shown in the sixth and seventh columns of Table 3. The total number of function evaluations and iterations is also listed. With an extension of the categorical parameter space, the GMVPS algorithm can be used for the optimization of the network topology with respect to the transfer function, which will be shown in the following section.

In the first numerical experiment, the network is restricted to at most five nonlinear hidden layers and a maximum of four neurons per layer. The

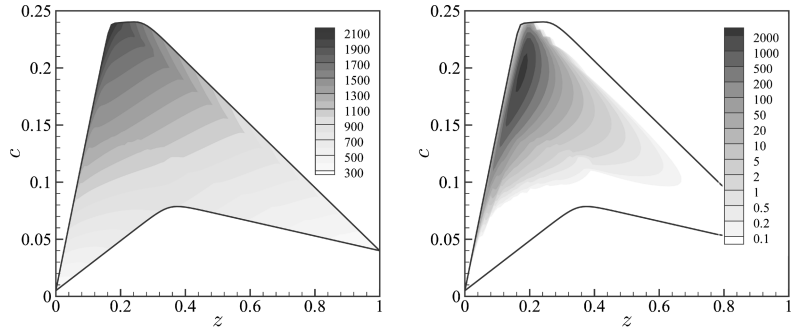
optimal network topology is obtained after 43 function evaluations and six iterations. In these iterations, two discrete poll steps and one continuous poll step were successful. The cost function of the optimal network is $J(\mathcal{A}) = -3.889$. The connectivity of the optimal network consists of a fully connected feedforward network in which all nonlinear hidden layers are connected to the input layer. It is interesting to note that the cost function of the fully connected feedforward network with the same nodal arrangement is $J(\mathcal{A}) = -2.574$. This fitness characteristic is very close to the fully connected feedforward 4-4-4-4-4-1 ANN. The better performance of the optimal network can be attributed to the following reasons. The increase of nodal connections results in an increase in the number of freely adjustable synaptic weights. A second reason for the better fitness might also be attributed to the direct connection of all nodes to the input. Because of this "shortcut," part of the partitioning of the input space is then delegated and distributed among all neurons in the hidden layers.

In the next example, the number of hidden layers is kept identical to the previous case; however, the maximum number of neurons per layer is doubled. The GMVPS algorithm requires 28 function evaluations and four iterations to find the network with the mesh local optimal fitness. Contrary to the previous example, in which nonlinear hidden layers are connected to the input layer, in this case only the second hidden layer has an additional link to the input layer. This additional connection decreases the cost function by 14 % compared to a fully connected feedforward 6-4-8-1 network. Note also that two of the hidden layers are empty. Interestingly, this example shows also that an improvement in the fitness is obtained even though the number of synaptic weights (103) and number of hidden layers is smaller than the topology in the previous example (121 synaptic weights).

The last row in Table 3 shows an example for a network with three nonlinear hidden layers and a maximum number of 16 neurons per layer. The network with optimal fitness was found after 28 function evaluations and seven iterations, of which two search steps and one continuous poll step were successful. The cost function is $J(\mathcal{A}) = -4.303$, and the connectivity resembles a fully connected feedforward network.

4.4 Chemistry Example. In this section, optimal neural networks for the approximation of a chemical system are generated. In this application, all three optimization strategies—optimal nodal arrangement, transfer functions, and connectivity—are combined.

The chemical process under consideration describes methane/air combustion. The GRI 2.11 chemical mechanism (Bowman et al., 1997) containing 277 elementary chemical reactions among 49 species is used. The steady laminar flamelet equations (Peters, 1984) are often employed to describe the reaction-diffusion balance in nonpremixed flames. The solutions to these equations provide temperature and mass fractions of all species in terms of



(a) Temperature [K] (b) Chemical source term [kg/(m^{3}s)]

Figure 8: Solution of the steady laminar flamelet equations as a function of mixture fraction z and progress variable c. (a) Temperature. (b) Chemical source term.

two parameters. The mixture fraction z and the reaction progress variable c are used for this parameterization. Temperature and the chemical source term of c, which can be viewed as a measure of heat release, are shown as functions of these parameters in Figure 8. In the following, network optimization for the approximation of these two quantities is performed. The solid lines represent the system boundary, and flame states outside this boundary are unaccessible. It can be seen that the chemical source term is very localized around $z \approx 0.2$ and $0.15 \leq c \leq 0.23$.

The number of samples used during the training phase and testing process is 500. Sample data are obtained by applying an acception-rejection method (Rubinstein, 1981) that results in higher sampling density in regions where the functional value Y is large. This method consequently results in a better resolution of the important regions with high chemical source term and temperature.

By considering the optimization of the nodal transfer function and the connectivity in the network design, the categorical neighborhood requires an extension in order to identify a network with optimal fitness in the joint continuous and categorical space. In this example, the categorical neighborhood for the transfer function includes the following combinations:

- The transfer functions for the neurons in all layers are identical, of either type T1 or T2.
- Except for the neurons in one layer, the transfer function for the neurons is identical.

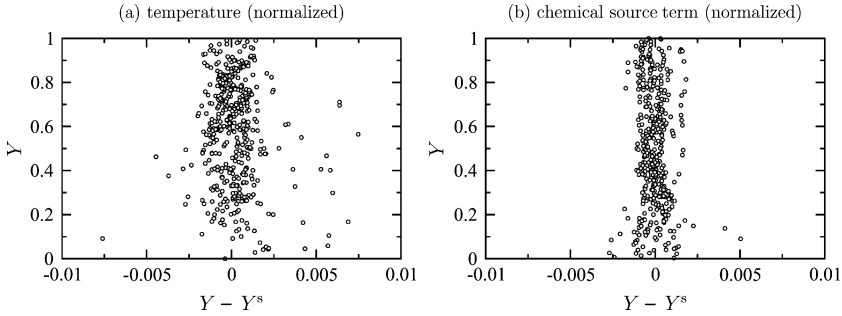


Figure 9: Regression analysis for approximation of (a) temperature and (b) chemical source with optimal network architecture. Note that the data for temperature and chemical source term are shown in nondimensional form.

The set of neighbors for the connectivity contains the following combinations:

- Fully connected feedforward network
- Fully connected feedforward network with $L_{i0} = 1$ where i represents any hidden layer
- Fully connected feedforward network with $L_{i0} = 1$ for all i hidden layers

The poll trigger is set to $\xi = 0.05$, and the maximum number of neurons per layer and the number of nonlinear hidden layers are set to eight and five, respectively.

The optimal network for the approximation of the temperature consists of a 6-7-4-7-1 ANN. All neurons employ a cyclometric transfer function. The connectivity resembles a fully connected feedforward network in which, in addition, all neurons are directly connected to the input layer. The cost function of this network is $J(\mathcal{A}) = -2.780$. The regression analysis for the network is shown in Figure 9a.

The network for the approximation of the chemical source term is considerably different from that of the temperature. This is mainly attributed to the different shapes of the functions for temperature and chemical source term. The architecture of the optimal network consists of a 2-2-8-8-1 fully connected feedforward network. The cost function for this network is $J(\mathcal{A}) = -3.124$, and the regression analysis for this network is shown in Figure 9b. Both network characteristics are summarized in Table 4.

It is interesting to compare the network performance with a conventional tabulation technique. Here, a particular chemistry table will be denoted by $\mathcal{T}$, and the parameter space of z and c will be discretized using a Cartesian and equidistant grid. The function to be tabulated is linearly interpolated to grid points. The performance for the tabulation of temperature and chemical

Table 4: Optimal Network Architecture for the Approximation of Temperature and Chemical Source Term.

Quantity	Neurons	Transfer Function	$\underline{\underline{L}}$	$J(\mathcal{A})$
Temperature	6-7-4-7-1	(T2)-(T2)-(T2)-(T2)-(T5)	$\begin{pmatrix} 0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0\\ 1&1&0&0&0&0&0\\ 1&0&1&0&0&0&0\\ 1&0&0&1&0&0&0\\ 1&0&0&0&1&0&0\\ 0&0&0&0&0&1&0 \end{pmatrix}$	−2.780
Chemical source term	2-2-8-8-1	(T2)-(T2)-(T2)-(T2)-(T5)	$\begin{pmatrix} 0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0\\ 0&1&0&0&0&0&0\\ 0&0&1&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&1&0&0\\ 0&0&0&0&0&1&0 \end{pmatrix}$	−3.124

Note: The transfer function and connectivity matrix are free categorical parameters.

Table 5: Comparison of the Performance for the Representation of Temperature and Chemical Source Term Using Conventional Tabulation Technique.

	Cost Function $J(\mathcal{T})$	
Table Size $N_z \times N_c$	Temperature	Chemical Source Term
100×100	−2.240	−1.677
200×200	−2.580	−2.193
300×300	−2.773	−2.492
400×400	−2.871	−2.621
500×500	−2.974	−2.791

source term with increasing resolution is presented in Table 5. In order to allow an objective comparison between the methods, the same set of test samples is used. Generally it can be observed that the performance of the tabulation method increases monotonically with finer resolution. The resolution required for an accurate chemistry representation grows rapidly and may impose a rather restrictive bound when the number of parameters is larger than three. It is interesting to point out that for an equivalent representation of the temperature using the optimal ANN described above, a table with about 300×300 grid points in z and c direction would be required. By comparing the fitness of the ANN and tabulation for the representation of the chemical source term (see Tables 4 and 5), it can be concluded that more than 500×500 grid points are required for a table to obtain an accuracy comparable with the network approximation.

5 Discussion

Optimization of an ANN has been successfully carried out using standard pattern search methods in conjunction with a surrogate function and generalized mixed variable pattern search. These methods offer extensive flexibility in implementation and freedom in choice of parameters. Using a model problem, optimization for the number of neurons, the transfer function, and the connectivity was explored. It was demonstrated that the use of categorical variables for the transfer function and connectivity resulted in the identification of better-performing ANNs compared with a trial-and-error approach.

Based on experience from the model problem, an ANN was optimized to model the chemistry of methane-air combustion for prediction of temperature and chemical source term. Regression analysis for the optimal ANN demonstrates satisfactory prediction of both quantities. It is interesting to note that the optimal ANNs for prediction of these two quantities are very different, which indicates underlying discrepancies in the behavior of each of these two quantities. Network performance is compared with the conventional tabulation method. It is shown that the chemistry tabulation requires considerably more memory in order to obtain equivalent accuracy.

The optimization methods presented here could be adapted in future work in several ways. First, it is possible to add constraints to either the GPS method or the GMVPS using a filter, as done by Abramson (2004) and Audet and Dennis (2004). Although the addition of a constraint on computational cost was considered in this work, it was felt that the bounds on the numbers of neurons and layers represented an implicit constraint so that an additional constraint was redundant.

In this work, the GMVPS algorithm is used as global optimization method and incorporates a second optimization program, the Levenberg-Marquardt algorithm, used for the adjustment of the synaptic weights. An interesting extension of this work would be to use the GMVPS method to simultaneously optimize the network architecture and the synaptic weights. The optimization of the weights can be accelerated by providing derivative information (Abramson, Audet, & Dennis, 2004).

The code for the generation of optimal neural networks is available from the first author.

Acknowledgments

We gratefully acknowledge funding by the US Department of Energy within the Advanced Simulation and Computing program. Helpful discussions with Masoud Aryanpour and Andrew M. Bradley are gratefully acknowledged. We also thank Mark Abramson and John E. Dennis, Jr., for guidance on the optimization methodology.

References

Abramson, M. A. (2004). Mixed variable optimization of a load-bearing thermal insulation system using a filter pattern search algorithm. *Optimization and Engineering, 5*(2), 157–177.

Abramson, M. A., Audet, C., & Dennis, Jr., J. E. (2004). Generalized pattern searches with derivative information. *Mathematical Programming, Series B, 100*, 3–25.

Angeline, P. J., Saunders, G. M., & Pollack, J. B. (1994). Evolutionary algorithm that constructs recurrent neural networks. *IEEE Trans. Neural Networks, 5*, 54–65.

Audet, C., & Dennis, Jr., J. E. (2000). Pattern search algorithms for mixed variable programming. *SIAM Journal on Optimization, 11*(3), 573–594.

Audet, C., & Dennis, Jr., J. E. (2003). Analysis of generalized pattern searches. *SIAM Journal on Optimization, 13*(3), 889–903.

Audet, C., & Dennis, Jr., J. E. (2004). A pattern search filter method for nonlinear programming without derivatives. *SIAM Journal on Optimization, 14*(4), 980–1010.

Blasco, J. A., Fueyo, N., Dopazo, C., & Ballester, J. (1999). Modelling the temporal evolution of a reduced combustion chemical system with an artificial neural network. *Combust. Flame, 113*(1–2), 38–52.

Blasco, J. A., Fueyo, N., Dopazo, C., & Chen, J. Y. (2000). A self-organizing-map approach to chemistry representation in combustion applications. *Combust. Theory Modelling, 4*(1), 61–76.

Blasco, J. A., Fueyo, N., Larroya, J. C., Dopazo, C., & Chen, J. Y. (1999). Single-step time-integrator of a methane-air chemical system using artificial neural networks. *Computers and Chemical Engineering, 23*(9), 1127–1133.

Booker, A. J., Dennis, Jr., J. E., Frank, P. D., Serafini, D. B., Torczon, V., & Trosset, M. W. (1999). A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization, 17*(1), 1–13.

Bornholdt, S., & Graudenz, D. (1992). General asymmetric neural networks and structure design by genetic algorithms. *Neural Networks, 5*(2), 327–334.

Bowman, C. T., Hanson, R. K., Davidson, D. F., Gardiner, W. C., Lissianski, V., Smith, G. P., Golden, D. M., Frenklach, M., & Goldenberg, M. (1997). GRI-Mech 2.11. Available online at http://www.me.berkeley.edu/gri-mech/.

Chen, J. Y., Blasco, J. A., Fueyo, N., & Dopazo, C. (2000). An economical strategy for storage of chemical kinetics: Fitting *in situ* adaptive tabulation with artificial neural networks. *Proc. Comb. Institute, 28*, 115–121.

Christo, F. C., Masri, A. R., & Nebot, E. M. (1996). Artificial neural network implementation of chemistry with PDF simulation of H_2/CO_2 flames. *Combust. Flame, 106*(4), 406–427.

Christo, F. C., Masri, A. R., Nebot, E. M., & Pope, S. B. (1996). An integrated PDF/neural network approach for simulating turbulent reacting systems. *Proc. Comb. Institute, 26*, 43–48.

Fausett, L. V. (1994). *Fundamentals of neural networks: Architectures, algorithms, and applications*. Englewood Cliffs, NJ: Prentice Hall.

Flemming, F., Sadiki, A., & Janicka, J. (2005). LES using artificial neural networks for chemistry representation. *Progress in Computational Fluid Dynamics, 5*(7), 375–385.

Frean, M. (1990). The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation, 2*(2), 198–209.

Hagan, M. T., Demuth, H. B., & Beale, M. (1996). *Neural network design*. Boston: PWS Publishing Company.

Haykin, S. (1994). *Neural networks: A comprehensive foundation*. Upper Saddle River, NJ: Prentice Hall.

Husken, M., Jin, Y., & Sendhoff, B. (2005). Structure optimization of neural networks for evolutionary design optimization. *Soft Computing, 9*(1), 21–28.

Hwang, M. W., Choi, J. K., & Park, J. (1997). Evolutionary projection neural networks. In *Proc. IEEE Conference on Evolutionary Computation (ICEC '97)* (pp. 667–671). Piscataway, NJ: IEEE.

Koehler, J. R., & Owen, A. B. (1996). Computer experiments. In S. Ghosh & C. Rao (Eds.), *Handbook of statistics* (pp. 261–308). New York: Elsevier Science.

Kokkolaras, M., Audet, C., & Dennis, Jr., J. E. (2000). *Mixed variable optimization of the number and composition of heat intercepts in a thermal insulation system* (Tech. Rep. TR00-21). Houston: Department of Computational and Applied Mathematics, Rice University.

Koza, J. R., & Rice, J. P. (1991). Genetic generation of both the weights and architecture for a neural network. In *Proc. IEEE Int. Joint Conf. Neural Networks (IJCNN'91 Seattle)* (pp. 397–404). Piscataway, NJ: IEEE.

Lam, S. H., & Goussis, D. A. (1988). Understanding complex chemical kinetics with computational singular perturbation. *Proc. Comb. Institute, 22*, 931–941.

Liu, Y., & Yao, X. (1996). Evolutionary design of artificial neural networks with different nodes. In *Proc. IEEE Conference on Evolutionary Computation (ICEC '96)* (pp. 670–675). Piscataway, NJ: IEEE.

Lophaven, S. N., Nielsen, H. B., & Søndergaard, J. (2002). *DACE: A Matlab Kriging toolbox version 2.0* (Tech. Rep. IMM-TR-2002-12). Copenhagen: Technical University of Denmark.

Lucidi, S., Piccialli, V., & Sciandrone, M. (2005). An algorithm model for mixed variable programming. *SIAM Journal on Optimization, 15*(4), 1057–1084.

Maas, U., & Pope, S. B. (1992). Simplifying chemical kinetics: Intrinsic low-dimensional manifolds in composition space. *Combust. Flame, 88*(3–4), 239–264.

Marsden, A. L., Wang, M., Dennis, Jr., J. E., & Moin, P. (2004). Optimal aeroacoustic shape design using the surrogate management framework. *Optimization and Engineering, 5*(2), 235–262.

McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics, 21*(2), 239–245.

Miller, G. F., Todd, P. M., & Hegde, S. U. (1989). Designing neural networks using genetic algorithms. In J. D. Schaffer (Ed.), *Proc. 3rd Int. Conf. Genetic Algorithms and Their Applications* (pp. 379–384). San Francisco: Morgan Kaufmann.

Mozer, M. C., & Smolensky, P. (1989). Skeletonization: A technique for trimming the fat from a network via relevance assessment. *Connection Sci., 1*(1), 3–26.

Peters, N. (1984). Laminar diffusion flamelet models in non-premixed turbulent combustion. *Prog. Energy Combust. Sci., 10*(3), 319–339.

Peters, N. (2000). *Turbulent combustion*. Cambridge: Cambridge University Press.

Pope, S. B. (1997). Computationally efficient implementation of combustion chemistry using in situ adaptive tabulation. *Combust. Theory Modelling, 1*(1), 41–63.

Rubinstein, R. Y. (1981). *Simulation and the Monte Carlo method*. New York: Wiley.

Serafini, D. B. (1998). *A framework for managing models in nonlinear optimization of computationally expensive functions*. Unpublished doctoral dissertation, Rice University.

Tang, K. S., Chan, C. Y., Man, K. F., & Kwong, S. (1995). Genetic structure for NN topology and weights optimization. In *Proc. 1st IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95)* (pp. 250–255). London: IEE.

Tonse, S. R., Moriarty, N. W., Brown, N. J., & Frencklach, M. (1999). PRISM: Piecewise reusable implementation of solution mapping. An economical strategy for chemical kinetics. *Israel Journal of Chemistry, 39*(1), 97–106.

Torczon, V. (1997). On the convergence of pattern search algorithms. *SIAM Journal on Optimization, 7*(1), 1–25.

Yao, X. (1999). Evolving artificial neural networks. *Proc. IEEE, 87*(9), 1423–1447.

Received August 16, 2006; accepted January 10, 2007.