Multi-network evolutionary systems and automatic problem decomposition

Vineet Khare, Xin Yao, Bernhard Sendhoff

2006

International Journal of General Systems
Vol. 00, No. 00, DD Month 200x, 1–14

Multi-network evolutionary systems and automatic decomposition of complex problems

VINEET R. KHARE†, XIN YAO*†, and BERNHARD SENDHOFF ‡
† CERCIA, School of Computer Science, The University of Birmingham, Birmingham B15 2TT, UK
‡ Honda Research Institute Europe GmbH, Carl-Legien-Straße 30, 63073, Offenbach/Main, Germany

(Received 00 Month 200x; In final form 00 Month 200x)

Multi-network systems, i.e., multiple neural network systems, can often solve complex problems more effectively than their monolithic counterparts. Modular neural networks tackle a complex problem by decomposing it into simpler sub-problems and then solving them. Unlike the decomposition in modular neural networks, a neural network ensemble usually includes redundant component nets and is often inspired by statistical theories. This paper presents different types of problem decompositions and discusses the suitability of various multi-network systems for different decompositions. A classification of various multi-network systems, in the context of problem decomposition, is obtained by exploiting these differences. Then a specific type of problem decomposition, which gives no information about the sub-problems and is often ignored in literature, is discussed in detail and a novel modular neural network architecture for problem decomposition is presented. Finally, a co-evolutionary model is presented, which is used to design and optimize such modular neural networks with sub-task specific modules. The model consists of two populations. The first population consists of a pool of modules and the second population synthesizes complete systems by drawing elements from the pool of modules. Modules represent a part of the solution, which co-operate with each other to form a complete solution. Using two artificial supervised learning tasks, constructed from smaller sub-tasks, it can be shown that if a particular task decomposition is better than others, in terms of performance on the overall task, it can be evolved using the co-evolutionary model.

Keywords: Multi-network Systems, ensembles, modular neural networks, co-evolution.

1 Introduction

Decomposing a complex computational problem into sub-problems, which are computationally simpler to solve individually and which can be combined to produce a solution to the full problem, can efficiently lead to compact and general solutions. Ideally for a good decomposition, these sub-problems will be much easier than the corresponding monolithic problem. In most cases such a decomposition relies on human experts and domain analysis. A system that can produce modules, which solve a subset of a big problem, can save us from manually crafting them. Also it can help discover potentially useful but unintuitive decompositions overlooked by humans. Ideally, both - the number of modules and the role that each one plays in the solution, should emerge automatically within the system. Many problems can only be decomposed into interdependent subcomponents hence changes in the role of one subcomponent effect the others. So the solutions should co-adapt and collectively solve the problem. Co-evolution is well suited for modelling the

*Tel.:+44-121-414-3747. E-mail: x.yao@cs.bham.ac.uk interdependencies among the subcomponents of the system and has been used in the literature to implement the *divide-and-conquer* strategy for tackling complex computational problems. These co-evolutionary methods can be subdivided further into two categories – single and two-level co-evolutionary methods. In single level co-evolutionary methods (Potter and DeJong 2000, Yong and Miikkulainen 2001) the sub-components/modules are evolved in separate genetically isolated sub-populations and fitness evaluations for these individuals are carried out by combining representative individuals from these subpopulations and then passing back the fitness of the system, thus created, to the representative individual. In two-level co-evolutionary methods (Moriarty and Miikkulainen 1997, García-Pedrajas *et al.* 2002, Khare *et al.* 2004, García-Pedrajas *et al.* 2005) modules and complete systems are co-evolved in two separate populations.

This work concerns with the problem decomposition aspects of modular neural networks. It presents a taxonomy of literature on problem decomposition in machine learning and illustrates the correspondence between problem decomposition and multiple neural network or multi-network systems. It also presents a novel modular neural network architecture suitable for a very generic type of problem decomposition, often ignored in literature and describes a two-level co-evolutionary method to design and optimize these modular neural networks which have sub-task specific modules. A module represents a part of the solution, which co-operates with others in the module population to form a complete solution. Fitness of individuals in the module population is determined by their contribution towards various systems in system population. Evolutionary pressure to increase the overall fitness of the two populations provides the needed stimulus for the emergence of the sub-task specific modules.

The rest of the paper is organised as follows. In Sec. 2 a brief background on problem decomposition and its correspondence with multi-network systems is presented. Sec. 3 discusses modularity in artificial neural networks (ANNs) and its effects. Sec. 4 presents a two-level Radial Basis Function (RBF) Network architecture which is to be designed using the co-evolutionary model presented in sec. 5. Conclusions follow in sec. 6 with thoughts on usefulness of modularity in ANNs.

2 Background

2.1 *Problem decomposition in machine learning*

Divide-and-conquer strategy is often used for tackling complex computational problems. Problem decomposition involves dividing a problem into sub-problems, solving these separately and then combining the sub-problem solutions to obtain the solution to the original problem. In the context of machine learning, there are various ways in which this can be achieved. This section presents a taxonomy of literature on problem decomposition (fig. 1). In order to categorize existing literature on problem decomposition a few concepts are introduced. In *sequential decomposition* one tries to divide the overall learning task into steps. Mostly the first step involves an unsupervised learning phase which makes the subsequent supervised learning easier. *Parallel decomposition* involves dealing with sub-tasks simultaneously but separately e.g. the classic what-and-where vision task. Within parallel decomposition one finds *task-oriented* and *data-oriented* types of decomposition. In *task-oriented decomposition* one looks for decomposition cues in the application task itself. Tasks may consist of relatively independent sub-tasks. The extent of information available about the sub-tasks is crucial and can help in designing the solution to the problem. In *data-oriented decomposition*, data available for learning a problem is decomposed. A decomposition can be performed

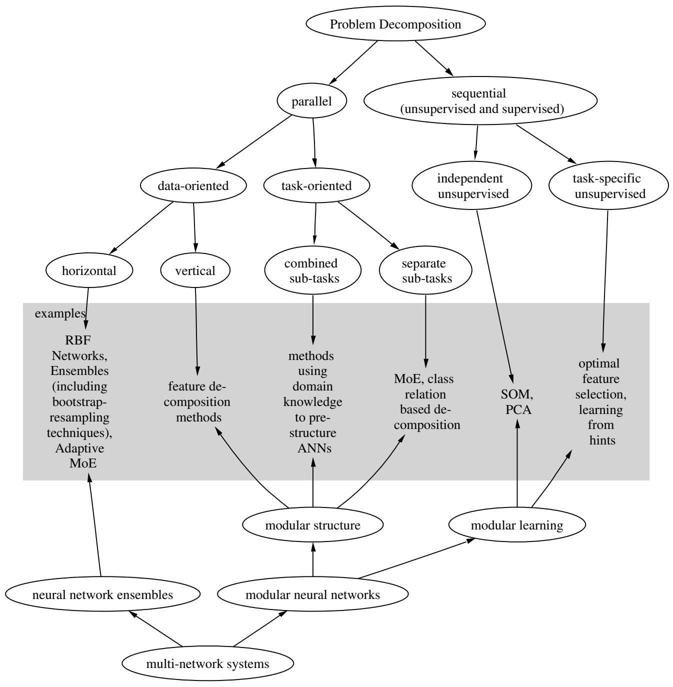

Problem Decomposition
parallel
sequential (unsupervised and supervised)
data-oriented
task-oriented
independent unsupervised
task-specific unsupervised
horizontal
vertical
combined sub-tasks
separate sub-tasks
examples
RBF Networks, Ensembles (including bootstrap-resampling techniques), Adaptive MoE
feature de-composition methods
methods using domain knowledge to pre-structure ANNs
MoE, class relation based de-composition
SOM, PCA
optimal feature selection, learning from hints
modular structure
modular learning
neural network ensembles
modular neural networks
multi-network systems

Figure 1. Correspondence between Problem Decomposition and Multi-network Systems.

on the input space (*horizontal*) or on the input variables (*vertical*) (Ronco *et al.* 1997). Examples of *horizontal input space decomposition* include RBF Networks, Neural Network Ensembles (including bootstrap re-sampling techniques) and Adaptive Mixture of Local Experts (Jacobs *et al.* 1991b). Examples of vertical input space decomposition include feature decomposition methods which use information theoretic measures to decompose the input variables into packs (Liao and Moody 1999, Maimon and Rokach 2002), without using the target variables.

Within *sequential decomposition* of a task into supervised and unsupervised learning phases one finds instances where unsupervised learning is independent of the supervised learning task to be followed. In general transformations of inputs or features are obtained to facilitate the supervised learning. Examples of *independent unsupervised* learning in *sequential decomposition* include topology-preserving maps, self-organization to visual features and principal component extraction (Hrycej 1992, chap. 4) and others (Yang and Moody 1999, Zaffalon and Hutter 2002). *Task-specific unsupervised learning* in *sequential decomposition* involves instances where unsupervised learning is performed with a bias in favour of the supervised learning to be followed. Examples include supervised feature discovery, optimal feature subset selection for a given learning algorithm (Kohavi and John 1997), learning from hints (Abu-Mostafa 1993) and others (Opitz 1999, Sindhwani *et al.* 2004).

In task-oriented parallel decomposition when the decomposition into sub-tasks is known, one can have *separate* feedback from each sub-task to design a solution to the problem, which is not possible when the sub-tasks are *mixed.* In literature, for *task-oriented parallel decomposition*, one usually finds the following:

- One sub-task at time t: $f^t(\vec{X},\vec{Y}) = f_1(\vec{X}) \quad OR \quad f_2(\vec{Y})$ (Jacobs *et al.* 1991a, Jordan and Jacobs

1995, Lu and Ito 1999)
- Sub-tasks on separate outputs: $\vec{f}(\vec{X},\vec{Y}) = \{f_1(\vec{X}), f_2(\vec{Y})\}$ (Jacobs *et al.* 1991b, Hüsken *et al.* 2002)
- Combination of sub-tasks at one output: $f(\vec{X},\vec{Y}) = g(f_1(\vec{X}), f_2(\vec{Y}))$ (Jenkins and Yuhas 1992, Lendaris and Mathia 1994),

where $\vec{X}$ and $\vec{Y}$ are subsets of the attributes of the problem, which may or may not be overlapping. In the first two instances the decomposition is known a priori and separate feedback is available to the learning system for *separate sub-tasks.* This can be used to embed a priori knowledge into the system and possibly train the modules independent of each other.

This work involves the third instance with much less knowledge about the problem (function g unknown). *Mixed sub-tasks task-oriented parallel decomposition* is arguably the most generic form of problem decomposition and has barely been mentioned in the neural network literature. Available literature on such type of problem decomposition relies heavily on the domain knowledge available. This work aims to develop a system which can perform this type of decomposition with least amount of domain knowledge.

2.2 *Multi-network systems*

Multiple neural networks are an obvious choice for various types of problem decompositions mentioned in sec. 2.1. In addition to providing performance improvement over single network solutions, they also have other advantages depending on the type of combination. Multi-Net systems can be divided into two main categories – neural network ensembles (NNEs) and modular neural networks (MNNs). A NNE is a collection of redundant neural networks working together and has primarily statistical motivations behind it. Each member of a NNE, which is capable of solving the complete task, outputs an estimation of the complete task starting from different training datasets and weight initializations, hence with an increase in the number of networks the estimated value gets closer and closer to actual value (Brown 2004). A NNE distributes the learning task among a number of experts, which in turn divides the input space into a set of subspaces. Ensembles provide better generalization abilities and robust solutions. In MNNs, unlike ensembles, each constituent network only performs a part of the overall task and all the networks are required to arrive at a solution to the task. These are used when a monolithic system is not able to perform the complete task (Sharkey 1997) or when there is an improvement in performance due to task decomposition. This improvement stems from the inherent separation of sub-tasks, which separates conflicting features that compromise the ability of a fully-connected network on the sub-tasks. Task decomposition provided by modular neural networks can also lead to the solution being easier to understand and modify. Also, a modular solution has fewer parameters which leads to better generalization abilities. One can classify MNNs into two categories – MNNs with modular learning and MNNs with modular structure. MNNs with modular learning are used to deal with *sequential problem decomposition* where learning is performed in steps by different networks. While MNNs with modular structure deal with *parallel problem decomposition* and sub-parts of the network solve the sub-problems simultaneously.

Redundancy and modularity are the two characteristics based on which one can differentiate between NNEs and MNNs (Sharkey 1999) but there are other multi-net architectures (Liu 1998, Hansen 2000) that blur the boundary between the two and it has been suggested that this differentiation is misleading (Brown 2004). Here this differentiation is made solely for the purpose of explaining the correspondence between problem decomposition and multi-network systems (fig. 1).

3 Modularity in ANNs

In sec. 2 we described how modular neural networks have been used in literature for problem decomposition. The concept of modularity though is a bit illusory. Again there are many ways in which modularity can be defined. A modularity measure can be derived from the connectivity within a neural network (structural) or from the (functional) decomposition they perform. Here a specific type of problem decomposition is dealt with (sec 3.1 presents one such representative problem). This section defines a functional modularity measure (sec. 3.2) for this problem and later explores how this measure is influenced by the type of network, mode of training and learning algorithm.

3.1 *Artificial problem: linear mixture of sub-tasks*

As an example of a problem requiring *task oriented parallel decomposition*, an artificial time series prediction task is constructed by combining two sub-tasks using a linear combination function. Mackey-Glass (MG) (Mackey and Glass 1977) and Lorenz (LO) (Lorenz 1963) time series prediction problems are used as the two sub-tasks. Detailed description of how these two sub-tasks are generated is omitted here for space constraints and can be found in Khare *et al.* (2005). These

Problem	**Inputs**						**Prediction Task**
Mackey-Glass	MG_3		MG_2		MG_1		MG_0
Lorenz-z		LO_3		LO_2		LO_1	LO_0
MG-LO	MG_3	LO_3)	MG_2	LO_2	MG_1	LO_1	g(MG_0, LO_0)

Table 1. Artificial time series mixture problems. $MG_x \equiv MG(t\text{-}x\Delta_1)$ and $LO_x \equiv LO(t\text{-}x\Delta_2)$ where $\Delta_1 = 1$ and $\Delta_2 = 0.02$ are the time steps used to generate the two time series.

two time series (MG and LO) are mixed according to Table 1 to create the mixture problems. Both sub-tasks involve prediction of the time series at time step t based on three previous time steps. For instance for Mackey-Glass task, MG(t) is to be predicted using MG(t-3Δ_1), MG(t-2Δ_1) and MG(t-Δ_1). The complete task is to predict g(MG(t), LO(t)). For all problems thus created the only feedback, the network (modular or not) gets, is its performance on the combined task (g). These two time series are relatively independent with a correlation coefficient of 0.032 for 1500 points and hence create mixture problems which favour a decomposition into independent modules. A linear (*Averaging*) combination problem is constructed by using g as averaging ($g(MG(t), LO(t)) = 0.5(MG(t) + LO(t))$).

3.2 *Functional measure of modularity*

Let C_{ij}^M and C_{ij}^L be the correlation coefficients between the output of hidden unit i in module j and the last (third) input of the Mackey-Glass and Lorenz time series, respectively. Last input is chosen because it is a good approximation to the next step which is to be predicted (sub-task). Modularity of the hidden unit i in module j, m_{ij} is defined as: $m_{ij} = \mid |C_{ij}^M| - |C_{ij}^L| \mid$.

Note that $-1.0 \leq C_{ij}^{M} \leq 1.0$ and $-1.0 \leq C_{ij}^{L} \leq 1.0$ imply $0.0 \leq m_{ij} \leq 1.0$. Modularity of the network is the average of modularities of all hidden units in all the modules : $M = \frac{1}{numMod}\sum_{j=1}^{numMod}\frac{1}{numUnit_j}\sum_{i=1}^{numUnit_j} m_{ij}$ and modularity measure for a fully-connected structure is : $M_{full} = \frac{1}{numUnit}\sum_{j=1}^{numUnit} m_j$. Full modularity is achieved at $M = 1.0$, where there is a complete separation of sub-tasks among hidden units.

3.3 *'Global' vs. 'clustering' neural networks*

The 'global' neural network as the MLP are characterised by the fact that all their nodes are involved to process each pattern. This is the difference with 'clustering' neural networks, as the RBF Networks, that process each pattern by involving only one part of their whole structure. This clustering of input space represents *horizontal data type decomposition* and to achieve problem decomposition automatically without much knowledge about the problem and the required decomposition RBF networks are used instead of MLPs. Since the overall task consists of relatively independent sub-tasks one would expect the decomposition along the number of patterns (horizontal) for one sub-task would be different from the other. So during training it can be assumed that some RBF units will specialize in MG task and others in LO task, thus also providing decomposition along the number of features (vertical). So the horizontal problem decomposition in RBF networks is expected to aid the vertical decomposition if the sub-tasks are relatively independent.

3.4 *Modularity vs. mode of training*

As seen in sec. 3.3, when both horizontal and vertical decompositions are performed simultaneously one aids another. This is also visible from figs. 2(a) and 2(b), which show the learning curves of an RBF network for the *averaging* problem. Here as the network learns the problem (horizontal decomposition; fig. 2(b)) its hidden units also specialize in the sub-tasks (vertical decomposition; fig. 2(a)). Fig. 2(a) shows the functional modularity index, for a RBF network, increasing with training epochs. As expected this is not the case with MLP training (fig. 2(c)).

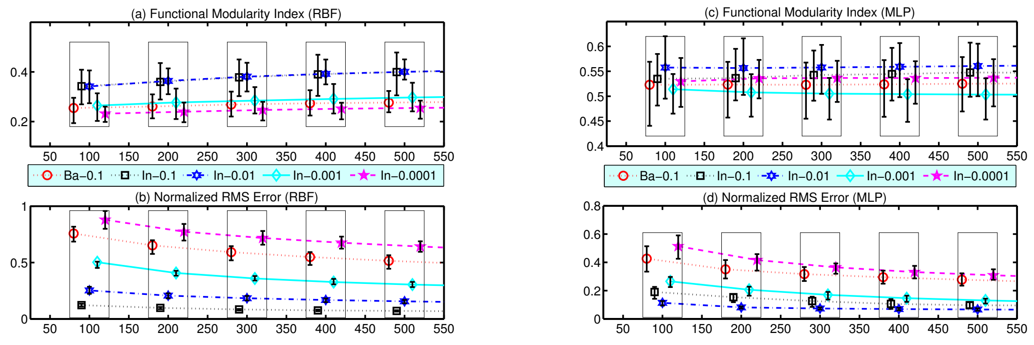


Figure 2. Learning curves with means, upper and lower quartiles (30 runs) at every 100 epochs of functional modularity indices (a and c) and normalized root-mean-squared errors (b and d) for the *Averaging* problem for a RBF (left) and a MLP (right) network. Ba $\equiv$ Batch and In $\equiv$ Incremental steepest descent.

Figs. 2(b) and 2(d) show the learning curves for incremental learning (learning rate $\eta = 10^{\alpha}$; $\alpha = -1, -2, -3$ and -4) and batch learning ($\eta = 10^{-1}$). As one moves towards lower and lower learning rates (i.e. towards batch learning) the extent of specialization (or functional modularity) reduces and for batch learning this value is very low, which also effects the overall learning performance of the network. This is because in batch learning the gradient calculation averages over all possible cases, hence correlations within a sub-problem are cancelled out.

3.5 *A modular solution to the problem*

Given the emergence of specialization among hidden units of the neural networks, one intuitive, and probably the optimal, solution for the linear mixture problem would be a modular neural network (fig. 3(a) and 3(b)) with modules solving the two sub-problems (*the pure-modular structure*, fig. 4(b)). In fig. 4 various networks represent four different decompositions for the combined problem. In order to validate the assumption that the pure-modular structure, being the intuitive decomposition for the problem, is the optimal one the learning curves of all these structures for the combined problem are compared. In addition, they are also compared with the *pre-trained pure-modular structure*. As the name suggests, modules in pre-trained pure-modular structure are trained separately from each other on individual sub-tasks. Since this structure has separate feedback available from all the modules (which is not the case with any other structure) and has pure modules, it can be used as a base case / ideal solution to the combination problem. These

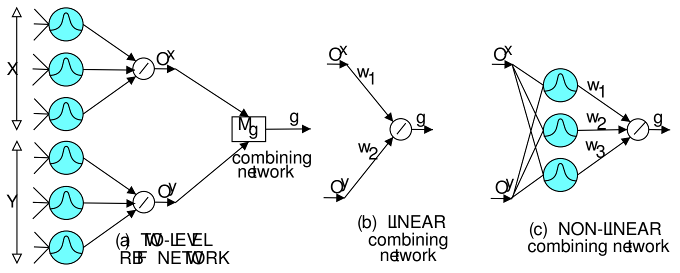

X
Y
O^x
O^y
M_g
g
combining network
(a) TWO-LEVEL RBF NETWORK
w_1
w_2
(b) LINEAR combining network
w_3
(c) NON-LINEAR combining network

Figure 3. A two-level (modular) RBF Network.

comparisons and the sub-task specialization of modules of the pure-modular structure into MG and LO sub-tasks, shown in figs. 5(a), 5(b) and 5(c), indicate that it represents a good decomposition of the task. It also fares well against the pre-trained pure-modular structure, but only a few other structures are tested and it can not be claimed that it is the optimal one. Optimality also depends on mode of training (sec. 3.4) and the learning algorithm (sec. 4.3) as the abilities of different algorithms to find a particular solution are different.

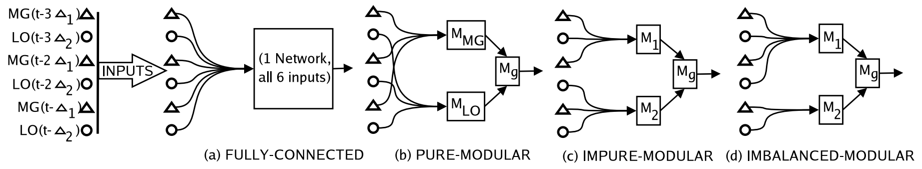


Figure 4. Four two-level RBF network structures representing four possible problem decompositions.

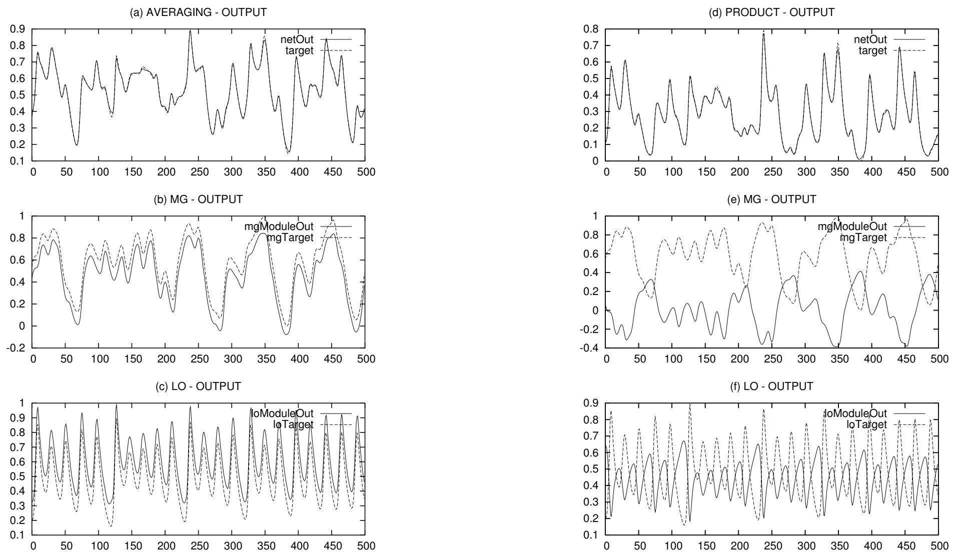


Figure 5. Sub-task specialization in two-level modular RBF network: Top two plots (a) and (d) show the output of the network against the target outputs of the *Averaging* and *Product* problems, respectively. Output of MG module and target output for MG sub-task in (b) and (d) show one module has specialized in MG sub-task for the *Averaging* and *Product* problems, respectively. Similarly, specialization to LO sub-task can be seen in (c) and (f) for the two problems. X-axis on each plot represents 500 test data points.

4 Two-level RBF network for automatic problem decomposition

In the last section the decomposition needed for the problem is only the decomposition of features into MG and LO subsets and the combination function is known. A more generic problem would be one where the combination function is unknown.

4.1 *Artificial problem: non-linear mixture of sub-tasks*

One can construct a non-linear combination problem by using g as a product ($g(MG(t), LO(t)) = MG(t) \cdot LO(t)$), or g as a whole-squared ($g(MG(t), LO(t)) = (MG(t) + LO(t))^2$) function in sec. 3.1. These are the *Product* and the *Whole-squared* problems. The pure-modular solution to these problems would be one with three modules. Two modules, M_{MG} and M_{LO}, that take inputs only from one source or, in other words, predict the output for a single time series and a third (M_g) that predicts their combination function (Fig. 4(b)). For the MG-LO problem, in Table 1, decomposition of features into Mackey-Glass and Lorenz is an example of *parallel decomposition* and calculation of MG(t) and LO(t) as an intermediate step while predicting g is an example of *sequential decomposition* (as discussed in Sec. 2.1).

4.2 *Two-level RBF network architecture*

The modular structure presented in sec. 3.5 can easily be extended to a Two-Level RBF Network capable of performing this parallel and sequential decomposition. For generic non-linear problems the outputs of the modules need to be combined non-linearly. For this purpose a RBF network as the *combining network* (Fig. 3(c)) is used. The sub-task specialization (figs. 5(d), 5(e), 5(f)) is also observed for this pure-modular two-level RBF network. Again the learning curves of various structures are given in fig. 4 (but with a non-linear combining module) are compared. For incremental learning pure-modular structure is observed to be better than other structures tested. This coupled with the sub-task specialization, indicate that the structure is capable of performing the sequential and parallel decompositions. There are two things that need be emphasized at this point, firstly best structure for the problem depends on the learning algorithm used for training (sec. 4.3) and secondly, in order for the modular structure to be beneficial all modules individually should be able to approximate their sub-tasks. If they are unable to approximate the sub-tasks properly the decomposition in terms of MG and LO modules would not be favoured.

4.3 *Modularity vs. learning algorithm*

In sec. 3.4, the effect of incremental and batch training on the learning capabilities of a fully-connected structure is observed. With batch training the network does not learn the *averaging* problem as well as it learns it with incremental training. Here different structures (fig 4) are compared for their learning capabilities with other sophisticated batch learning algorithms. *Improved Resilient-Backpropagation* (Rprop) (Igel and Hüsken 2003)), a *quasi-Newton method with BFGS update procedures* (see Bishop 1996, chap. 7) and $(1 + 1)$ Evolution Strategy as batch training algorithms, alongside the steepest descent algorithms, are used. Table 2 shows these comparative results.

Combination Function →	**Averaging**	**Product**	**Whole-squared**
Algorithms			
Steepest Descent			
(1) Incremental ($\eta = 0.1$)	Pu	Pu	Pu
(2) Batch ($\eta = 0.1$)	Pu	Fu	Fu
IRPROP	Pu	Fu	Fu
Quasi-Newton (BFGS)	Pu	Fu	Fu
Evolution Strategy	Pu	Fu	Fu

Table 2. After 100 epochs of training, multiple comparison of means using one-way anova results to determine which means (for 30 runs) are significantly different. Pu indicates that pure-modular structure is better than others and Fu indicates that either fully-connected structure is better than others or there is no single winner.

When the pure-modular structure is only required to perform parallel decomposition (*Averaging*) it easily achieves the performance of the pre-trained pure-modular (ideal) structure with all the algorithms used and emerges as a winner throughout. When both parallel and sequential decompositions are required it is only able to do that with the incremental version of steepest descent algorithm. Sec 5 presents a co-evolutionary model that can be used to evolve such a structure in a favourable learning environment, i.e. using incremental steepest descent algorithm.

5 Co-evolutionary modules and modular neural network (CoMMoN) model

A two level co-evolutionary architecture (fig. 6), with the lower level (level 1) as a population of modules and the higher level (level 2) as a population of complete systems made up of modules from module population, is used. Within this general co-evolutionary framework, both the structure and parameterization of the modules as well as the parameterization and structure of the systems can be evolved. Evolution at level 1 searches for good building blocks for the system and at level 2 it searches for good combinations of modules in level 1. This kind of co-evolutionary architecture has been used before (Moriarty and Miikkulainen 1997, García-Pedrajas *et al.* 2002, Khare *et al.* 2004) where neurons and artificial neural networks were used as modules and systems, respectively. In Khare *et al.* (2005) it is extended up a level to use networks and their combinations (two-level RBF networks) as the lower and the higher levels, respectively, for the non-linear combination problems.

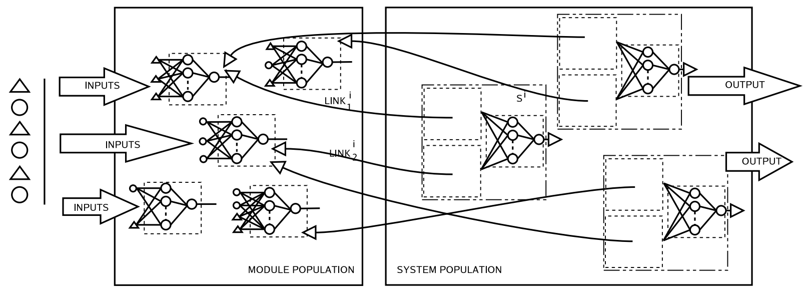


Figure 6. Co-evolutionary Model for the non-linear combination problem. Each individual in module population is a RBF network and each individual in system population contains a RBF network and pointers to two modules in module population.

For the non-linear combination (*Product*) problems, modules are RBF networks (Fig. 6) differing from each other in terms of the inputs that they get out of all six inputs from the combined problem, hence each module can have between one and six inputs. Each system contains pointers to two modules in module population and the output of the system is obtained with the help of *combining network*, with two inputs and one output. Systems are trained using incremental steepest descent learning, with random parameter initializations, in each generation and, are assigned with a fitness value depending on their performance on validation data set. Modules derive their fitness from the systems to which they contribute in the system population. They are to be judged on the basis of their contribution towards the complete problem. Each module is assigned with a fitness value equal to the number of appearances it made in any system in system population, during the last 10 generations. A detailed description of the co-evolutionary model, experimentation with *Averaging* and *Product* problems and results can be found in Khare *et al.* (2005).

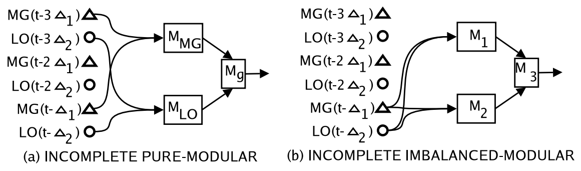


Figure 7. Two solutions for the *Product* problem from the co-evolutionary model.

If a pure modular structure is advantageous over others then some of the modules in module population will specialize in MG prediction task and some in LO prediction task or in other words starting from a completely random configuration it will converge to a state where some modules will take their inputs only from Mackey-Glass time-series and some from Lorenz time-series. The difference in performance between this structure and others provides the co-evolutionary model with the selection pressure towards this structure (problem decomposition). *Averaging* problem requires *parallel* (MG and LO modules) and *Product* problem requires both *parallel* and *sequential* (first evaluation MG and LO modules and then the product module) decompositions. For the *Averaging* problem the co-evolutionary model was able to discover the pure modular structure and for the *Product* problem in addition to the pure-modular structure it also discovered two other interesting structures (figs. 7(a) and 7(b)) very similar in performance to the pure-modular structure. Structure in fig. 7(a) indicates that all three inputs are not needed to solve the problem. Also, since it is known that for a time series prediction problem the last time step is the most important one, the structure in Fig. 7(b) represents another good solution to the problem where modules M_1 and M_2 are only focusing on the two most relevant inputs and the combination (M_3) is producing an *ensemble effect* of two very similar modules.

6 Conclusions

6.1 *When is modularity useful?*

Modularity has been recognised as one of the crucial aspects of development (embryogeny) and evolution (Schlosser and Wagner 2005). Likewise, modularity should also play an important role in designing artificial complex evolutionary and developmental systems. There has been some debate about whether one should build modularity into such systems. There are instances where modular architectures are shown to outperform fully-connected architectures primarily by minimizing modular interference. Arguments in favour often cite human brain (Rueckl *et al.* 1989, Ferdinando *et al.* 2001), which is a result of evolution by natural selection and has functionally specialised neural modules. Arguments against warn (Bullinaria 2002) that advantages of modularity are not as straightforward and by using efficient learning algorithms and sophisticated cost functions it is possible to deal with modular interference.

The answer to the question depends on how one defines being 'useful.' Experiments presented here indicate that the usefulness of modularity, if judged on the basis of the performance of the system on the overall task, depends on the learning algorithm used (as observed in Bullinaria (2002)) and by using a more sophisticated learning algorithm it is possible to achieve similar, if not better, performing non-modular structures as against modular structures. However, the performance measure is only tested in a static environment on a relatively low dimensional problem. Modularity in neural networks is expected to be much more beneficial in dynamic environments (Hüsken *et al.* 2000), where it can help improve the speed of learning.

Modularity can also prove useful in evolving highly complex phenotypes. One to one mapping of these to genotypes results in a very high dimensional genotypic search space, which makes the problem intractable. In nature this problem is tackled through gene duplication (independent specialization of duplicate genes), which corresponds to an inherent modularisation of the system development. This concept has been used in artificial evolutionary systems for indirect genotype-phenotype mapping (Gruau *et al.* 1996, Calabretta *et al.* 1998) to solve the intractability problem. Further in evolution, the genetic representation encodes a growth process, the organization of such a process inherently favours modular systems (Sendhoff and Kreutz 1999). Modularity might be a consequence of this process. However, if one does not believe that this assumingly 'simple' consequence is an accident, then there must be a selective advantage in this process and in turn the easy design of a modular system might be one such advantage.

6.2 *Modularity can be evolved*

Given that modularity is beneficial for the overall performance of the system, in this case with incremental steepest descent learning, it can be evolved. This work has presented a two-level co-evolutionary model to design and optimize modular neural networks with sub-task specific modules. The first level population consists of a pool of modules and the second level synthesizes systems by drawing elements from this pool. Modules represent parts of the solution, which co-operate with others in the module population to form a complete solution. Fitness of individuals in the module population is determined by their contribution towards various systems in system population. Evolutionary pressure to increase the overall fitness of the two populations provides the needed stimulus for the emergence of the sub-task specific modules. With the help of artificial tasks created by mixing two sub-tasks (MG and LO time series prediction) it is demonstrated that if a particular task decomposition is better in terms of performance (mean-squared-error in this case) on the overall task it can be evolved using this co-evolutionary model. This is also evident from the emergence of some good decompositions which one would not think of while designing such a modular system manually.

6.3 *Future work directions*

It would interesting to investigate the effects of modularity in a dynamic environment (sec. 6.1). Within the static framework the co-evolutionary model has its share of limitations as well. Firstly, the number of modules is fixed and is supplied to the model as a priori knowledge and secondly, within these modules the structure of the module which combines the other modules to produce the final output is also fixed. Ideally, and in line with the idea of co-adapting all the modules together, the structure for this module (*combining network*) should also be evolved along with other modules. Adaptation of the number of modules in a system can be achieved in two different ways. First one is a stage-wise approach, where new modules are added or deleted simultaneously in all the systems present in the population. This happens only when there is a stagnation in performance of systems in system population. A more global approach is where systems with different number of modules are present at the same time in the population. To overcome the second limitation the model can be extended to co-evolve the modules for combining other modules alongside those other modules in the module population. Finally, this generic model can be applied to a variety of problems ranging from feature decomposition and feature selection in neural network ensembles to problems which require pre-processing. Feature decomposition can be viewed as an example of parallel decomposition and, feature selection and pre-processing can be viewed as an example of first stages in a multi-stage sequential decomposition problem.

References

Y. S. Abu-Mostafa, "A method for learning from hints," in S. J. Hanson, J. Cowan and C. L. Giles (eds.), *Advances in Neural Information Processing Systems*, vol. 5, pp. 73–80, San Mateo, CA, Morgan Kaufmann, (1993).

C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford, UK: Oxford University Press, (1996).

G. Brown, "Diversity in Neural Network Ensembles," Ph.D. thesis, School of Computer Science, University of Birmingham, (2004).

J. A. Bullinaria, "To modularize or not to modularize?" in J. Bullinaria (ed.), *Proceedings of the 2002 U.K. Workshop on Computational Intelligence (UKCI-02)*, pp. 3–10, Birmingham, (2002).

R. Calabretta, S. Nolfi, D. Parisi and G. P. Wagner, "A case study of the evolution of modularity: Towards a bridge between evolutionary biology, artificial life, neuro- and cognitive science," in C. Adami, R. K. Belew, H. Kitano and C. E. Taylor (eds.), *Proceedings of the Sixth International Conference on Artificial Life (ALIFE VI)*, pp. 275–284, University of California, Los Angeles, MIT Press, (1998).

A. D. Ferdinando, R. Calabretta and D. Parisi, "Evolving modular architectures for neural networks," in R. M. French and J. P. Sougné (eds.), *Proceedings of the Sixth Neural Computation and Psychology Workshop*, pp. 253–264, Liege, Belgium, Springer Verlag, (2001).

N. García-Pedrajas, C. Hervás-Martínez and J. Muñoz-Pérez, "Multi-objective cooperative coevolution of artificial neural networks," *Neural Networks*, 15, 1259–1278, (2002).

N. García-Pedrajas, C. Hervás-Martínez and D. Ortiz-Boyer, "Cooperative coevolution of artificial neural network ensembles for pattern classification," *IEEE Transactions on Evolutionary Computation*, 9, 271–302, (2005).

F. Gruau, D. Whitley and L. Pyeatt, "A comparison between cellular encoding and direct encoding for genetic neural networks," in J. R. Koza, D. E. Goldberg, D. B. Fogel and R. L. Riolo (eds.), *Genetic Programming 1996: Proceedings of the First Annual Conference*, pp. 81–89, Stanford University, CA, USA, MIT Press, (1996).

J. V. Hansen, "Combining predictors: Meta machine learning methods and bias/variance and ambiguity decompositions," Ph.D. thesis, Aarhus Universitet, Datalogisk Institut, (2000).

T. Hrycej, *Modular Learning in Neural Networks. A Modularized Appproach to Neural Network Classification*, New York: Wiley, (1992).

M. Hüsken, J. E. Gayko and B. Sendhoff, "Optimization for problem classes - neural networks that learn to learn," in X. Yao and D. F. Fogel (eds.), *2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks (ECNN 2000)*, pp. 98–109, New York, IEEE Press, (2000).

M. Hüsken, C. Igel and M. Toussaint, "Task-dependent evolution of modularity in neural networks," *Connection Science*, 14, 219–229, (2002).

C. Igel and M. Hüsken, "Empirical Evaluation of the Improved Rprop Learning Algorithm," *Neurocomputing*, 50, 105–123, (2003).

R. A. Jacobs, M. I. Jordan and A. G. Barto, "Task Decomposition Through Competition in a Modular Connectionist Architecture: The What and Where Vision Tasks," *Cognitive Science*, 15, 219–250, (1991a).

R. A. Jacobs, M. I. Jordan, S. J. Nowlan and G. E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, 3, 79–87, (1991b).

R. E. Jenkins and B. P. Yuhas, "A simplified neural-network solution through problem decomposition: The caseof the truck backer-upper," *Neural Computation*, 4, 647–649, (1992).

M. I. Jordan and R. A. Jacobs, "Modular and hierarchical learning systems," in M. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, pp. 579–582, Cambridge, MA, MIT Press, (1995).

V. R. Khare, X. Yao and B. Sendhoff, "Credit Assignment among Neurons in Co-evolving Populations," in X. Y. et al. (ed.), *8th International Conference on Parallel Problem Solving from Nature, PPSN VIII*, pp. 882–891, Birmingham, UK, Springer. Lecture Notes in Computer Science. Volume 3242, (September 2004).

V. R. Khare, X. Yao, B. Sendhoff, Y. Jin and H. Wersing, "Co-evolutionary Modular Neural Networks for Automatic Problem Decomposition," in *(To appear) 2005 IEEE Congress on Evolutionary Computation, CEC 2005*, Edinburgh, UK, (September 2005).

R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, 97, 273–324, (1997).

G. G. Lendaris and K. Mathia, "On prestructuring anns using a priori knowledge," in *Proceedings of World Conference on Neural Networks (WCNN'94)*, Earlbaum/INNS, (June 1994).

Y. Liao and J. Moody, "Constructing heterogeneous committees using input feature grouping: Application to economic forecasting," *Advances in Neural Information Processing Systems*, 12, 921–927, (1999).

Y. Liu, "Negative correlation learning and evolutionary neural network ensembles," Ph.D. thesis, University College, The University of New South Wales, Australian Defence Force Academy, Canberra, Australia, (1998).

E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of atmospheric Science*, 20, 130–141, (1963).

B.-L. Lu and M. Ito, "Task decomposition and module combination based on class relations: A modular neural network for pattern classification," *IEEE Transactions on Neural Networks*, 10, 1244–1256, (1999).

M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, 197, 287–289, (1977).

O. Maimon and L. Rokach, "Improving supervised learning by feature decomposition," in *Proceedings of the Second International Symposium on Foundations of Information and Knowledge Systems, Lecture Notes in Computer Science*, pp. 178–196, springer, (2002).

D. E. Moriarty and R. Miikkulainen, "Forming Neural Networks Through Efficient and Adaptive Coevolution," *Evolutionary Computation*, 5, 373–399, (1997).

D. Opitz, "Feature selection for ensembles," in *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI)*, pp. 379–384, (1999).

M. A. Potter and K. A. DeJong, "Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents," *Evolutionary Computation*, 8, 1–29, (2000).

E. Ronco, H. Gollee and P. Gawthrop, "Modular neural networks and selfdecomposition," Tech. Rep. CSC-96012, Center for System and Control, University of Glasgow, Glasgow, UK, (1997).

J. G. Rueckl, K. R. Cave and S. Kosslyn, "Why are "What" and "Where" Processed by Separate Cortical Visual Systems? A Computational Investigation," *Journal of Cognitive Neuroscience*, 1, 171–186, (1989).

G. Schlosser and G. P. Wagner (eds.), *Modularity in Development and Evolution*, The University of Chicago Press, (2005).

B. Sendhoff and M. Kreutz, "Variable encoding of modular neural networks for time series prediction," in V. Porto (ed.), *Congress on Evolutionary Computation CEC*, pp. 259–266, IEEE Press, (1999).

A. J. C. Sharkey (ed.), *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, Secaucus, NJ, USA: Springer-Verlag New York, Inc., (1999).

N. E. Sharkey, "Neural networks for coordination and control: The portability of experiential representations," *Robotics and Autonomous Systems*, 22, 345–359, (1997).

V. Sindhwani, S. Rakshit, D. Deodhare, D. Erdogmus, J. Principe and P. Niyogi, "Feature Selection in MLPs and SVMs based on Maximum Output Information," *IEEE Transactions on Neural Networks*, 15, 937–948, (July 2004).

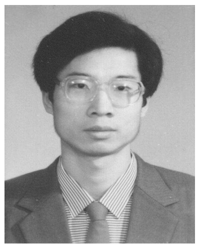
H. Yang and J. Moody, "Feature selection based on joint mutual information," in *Advances in Intelligent Data Analysis (AIDA), Computational Intelligence Methods and Applications (CIMA), International Computer Science Conventions*, pp. 22–25, Rochester, New York, (June 1999).

C. H. Yong and R. Miikkulainen, "Cooperative Coevolution of Multi-Agent Systems," Tech. Rep. AI01-287, Department of computer Sciences, The University of Texas at Austin, Austin, TX 78712 USA, (2001).

M. Zaffalon and M. Hutter, "Robust feature selection by mutual information distributions," in *Proc. of the 18th Conf. on Uncertainty in Artificial Intelligence*, pp. 577–584, San Francisco, Morgan Kaufmann, (2002).

Vineet R. Khare received his B. Tech degree in Mechanical Engineering from IIT Kanpur, India, in 2001 and his MSc degree in Natural Computation from The University of Birmingham, UK, in 2002. He is currently pursuing his doctoral studies in computer science at the University of Birmingham, UK.

He is a member of the Natural Computation Research Group, School of Computer Science, University of Birmingham. His research interests include co-evolution, neural network ensembles, modular neural networks and multi-objective evolutionary algorithms.



Xin Yao is a professor of computer science from the University of Birmingham, UK. He is also a Distinguished Visiting Professor of the University of Science and Technology of China (USTC), Hefei, China, and a visiting professor of three other universities. He is an IEEE Fellow, the Editor-in-Chief of IEEE Transactions on Evolutionary Computation, an associate editor or an editorial board member of ten other journals, the editor of the World Scientific book series on "Advances in Natural Computation", and a guest editor of several journal special issues. He won the 2001 IEEE Donald G. Fink prize paper award and several other best paper awards. He has been invited to present 35 keynote or plenary speeches at conferences worldwide. He has more than 200 research publications. His research interests include evolutionary computation, neural network ensembles, global optimization, data mining, computational time complexity of evolutionary algorithms, and real-world applications. In addition to basic research, he works closely with many industrial partners on various real-world problems. He is the Director of The Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA) (www.cercia.com), which is focused on applied research and knowledge transfer to the industry.

Dr. Yao received his BSc in 1982 (USTC), MSc in 1985 (North China Institute of Computing Technology) and PhD in 1990 (USTC), all in computer science. He had worked in China and Australia before joining the University of Birmingham in 1999 as a professor of computer science.

Bernhard Sendhoff studied physics at the Ruhr-Universität Bochum, Germany, and the University of Sussex, U.K. In 1993, he received the Diploma degree and in 1998 the Doctorate degree in physics from the Ruhr-Universität Bochum, Germany. He is currently Chief Technology Officer of the Honda Research Institute Europe GmbH, Offenbach/Main, Germany, in addition to being Head of the Evolutionary and Learning Technology Group. From 1994 to 1999, he was a researcher and postdoctorate member of the Institute of Neuroinformatik, Ruhr-Universität Bochum. His current research interests include the design and structure optimization of complex and adaptive systems based on neural networks, fuzzy systems, and evolutionary algorithms. He has published over 70 scientific papers and holds several patents.

Dr. Sendhoff is a senior member of the IEEE and a member of the ENNS and the DPG.