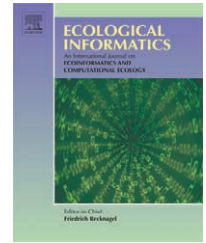


available at [www.sciencedirect.com](http://www.sciencedirect.com)[www.elsevier.com/locate/ecoinf](http://www.elsevier.com/locate/ecoinf)

# Current developments and future directions of bio-inspired computation and implications for ecoinformatics

Xin Yao\*, Yong Liu, Jin Li, Jun He, Colin Frayn

The Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, The University of Birmingham, Edgbaston, Birmingham B15 2TT, UK

## ARTICLE INFO

### Article history:

Received 15 July 2005

Accepted 15 July 2005

### Keywords:

Evolutionary computation  
Neural network ensembles  
Global optimisation  
Evolutionary neural networks  
Evolutionary algorithms  
Genetic algorithms

## ABSTRACT

Evolutionary and neural computation has been used widely in solving various problems in biological ecosystems. This paper reviews some of the recent work in evolutionary computation and neural network ensembles that could be explored further in the context of ecoinformatics. Although these bio-inspired techniques were not developed specifically for ecoinformatics, their successes in solving complex problems in other fields demonstrate how these techniques could be adapted and used for tackling difficult problems in ecoinformatics. Firstly, we will review our work in modelling and model calibration, which is an important topic in ecoinformatics. Secondly one example will be given to illustrate how coevolutionary algorithms could be used in problem-solving. Thirdly, we will describe our work on neural network ensembles, which can be used for various classification and prediction problems in ecoinformatics. Finally, we will discuss ecosystem-inspired computational models and algorithms that could be explored as directions of future research.

© 2005 Elsevier B.V. All rights reserved.

## 1. Introduction

Many algorithms in evolutionary computation and neural networks have been used in ecoinformatics. However, there have been some recent new developments that could be very useful for solving complex ecoinformatics problems. This paper reviews and summarises some of the recent work in evolutionary computation and neural network ensembles that are particularly relevant to ecoinformatics.

Data-driven modelling is a major topic of research in ecoinformatics, where the best model is to be found given a set of empirical data. Although it is not new, data-driven modelling is still an unsolved problem especially for noisy data. There are statistical and neural network approaches that could potentially help in modelling. However, one of the drawbacks for approaches like neural networks is the lack of

good comprehensibility of the results. It is often very hard to understand and interpret the neural network results, unless additional techniques such as rule extraction were used. Evolutionary computation techniques, e.g., genetic programming, offer opportunities to evolve model structures as well as model parameters explicitly. Their results are more comprehensible and easier to interpret than those results generated by the blackbox methods (e.g., neural networks). We will give one example in astrophysics to illustrate how data-driven modelling could be done by a two stage evolutionary approach (Li et al., 2004).

Coevolution is a typical phenomenon in biological ecosystems. Its idea can be used to design very effective problem-solving algorithms. The key idea behind coevolutionary algorithms is that the fitness of an individual depends on other individuals in the same population. The same individual may

\* Corresponding author.

E-mail address: [x.yao@cs.bham.ac.uk](mailto:x.yao@cs.bham.ac.uk) (X. Yao).

obtain very different fitness values in different populations. Coevolution has been studied extensively in recent years as a way to decomposing a complex problem automatically (Darwen and Yao, 1997; Khare et al., 2004; Liu et al., 2001). Given a complex problem, the idea here is to use coevolution to form species automatically, where each species is specialised in solving only parts of the entire problem. The entire problem is then solved by combining all different species together. This can be regarded as an automatic divide-and-conquer approach to complex problem solving. We will give one example in optimisation in this paper to illustrate how problem decomposition could be beneficial (Liu et al., 2001).

Another example of automatic problem decomposition is the neural network ensembles that we have proposed (Monirul Islam et al., 2003; Liu et al., 2000; Chandra and Yao, 2004). Instead of designing and training a monolithic neural network for a large and complex problem, a group of simpler neural networks were constructed and trained. Negative correlation and coevolution have been used to increase the diversity within the ensemble of neural networks and improve the generalisation ability of the entire neural network ensemble. Excellent results have been reported in the literature (Monirul Islam et al., 2003; Liu et al., 2000; Chandra and Yao, 2004).

The work introduced so far can be regarded as the application of bio-inspired approaches (e.g., evolutionary computation and neural network ensembles) to ecoinformatics problems, where the information flow appears to be from computational techniques to problems related to ecosystems. However, there is an opposite direction of information flow, where some ideas and concepts in biological ecosystems have inspired researchers to explore new forms of computation and new algorithms for problem solving. This is a very important part of ecoinformatics that is underexplored and understudied at the moment. We hope our paper will encourage more researchers in moving into these new research areas, which require close collaboration between computer scientists and ecologists.

The rest of this paper will be organised as follows. Section 2 discusses our modelling work using the two stage evolutionary approaches (Li et al., 2004). Section 3 describes our work on coevolutionary optimisation (Liu et al., 2001). Section 4 presents neural network ensembles and negative correlation learning (Chandra and Yao, 2004). Section 5 outlines the future research directions for possible ecosystem-inspired computational models and algorithms. Finally, Section 6 concludes the paper with a brief summary.

## 2. Data-driven evolutionary modelling

There are many approaches that one could use for data-driven evolutionary modelling (Li et al., 2002, 2004). We will review one example (Li et al., 2004) here where genetic programming was used to evolve model structures (i.e., explicit function forms) first. Then common “features” from evolved models were identified and abstracted. Finally, the abstracted model was calibrated by using evolutionary programming to fine-tune model parameters. More details about the example can be found in Li et al. (2004).

### 2.1. Distribution of light in elliptical galaxies (Li et al., 2004)

In optical images, galaxies have a wide range of shapes and sizes. To develop a deeper understanding of the evolution of galaxies, it is important to quantify their morphological features as a first stage in an attempt to link them to the physics of their formation.

The most commonly used fitting functions for elliptical galaxy profiles are the Hubble’s law given by

$$I(r) = \frac{I_0}{(r/a + 1)^2}, \quad (1)$$

and the de Vaucouleurs  $r^{1/4}$  law given by

$$I(r) = I_e \exp\{-3.33[(r/r_e)^{1/4} - 1]\}. \quad (2)$$

Each function has two parameters, a brightness scale  $I_0$  or  $I_e$  and a size scale  $a$  or  $r_e$ .  $I_0$  is the surface brightness at  $r=0$ , i.e., the central surface brightness. The scale length  $a$  is the distance at which the surface brightness falls to a quarter of its central value. In contrast,  $r_e$ , known as the effective radius, contains half of the total light of the galaxy, and  $I_e$  is the brightness at  $r_e$ . The two laws give a reasonably good description of the luminosity profiles of elliptical galaxies and the bulges of some types of spiral galaxies. However, they do not describe well these profiles beyond certain bounds of  $r$  or the bulges of spiral galaxies with pronounced spiral features. For the latter, a popular model is an exponential form

$$I(r) = 5.36I_e \exp[-1.68(r/r_e)]. \quad (3)$$

Neither Eq. (2) nor Eq. (3) has been formally linked to a physical model of galaxy formation, and they remain mere empirical fits to observed profiles. Hubble’s model (1) is not a very good fit to a wide range of observed ellipticals, but is related to possible density profiles of self-gravitating isothermal spheres of gas. It would be desirable to have a single model that would describe all elliptical galaxies with a variation of parameters, especially if this model could be linked with the physics of collapsing gas clouds and the formation of self-gravitating ellipsoidal systems.

There has been a lot of effort in the literature to find appropriate mathematical functions to describe a wide range of profiles of various components of spiral and elliptical galaxies, and to find plausible values for parameters that would fit observed profiles. Both tasks are non-trivial. The usual approach is to postulate a mathematical model comprising of common mathematical functions, then to apply fitting algorithms to find suitable parameters for these functions. The parameter fitting algorithm usually adopted is the non-linear reduced  $\chi^2$  minimisation given by

$$\chi^2 = \frac{1}{\nu} \sum_i \frac{[I_{\text{model}}(i) - I_{\text{obs}}(i)]^2}{\delta^2}, \quad (4)$$

where  $I_{\text{obs}}(i)$  is the individual observed profile value,  $I_{\text{model}}(i)$  is the value calculated from the fitting function,  $\nu$  is the number of degrees of freedom, and  $\delta$  is the standard deviation of the data. One disadvantage of non-linear minimisation algorithms is their sensitivity to the initial values provided.

Unreasonable initial values could cause the fitting program to trap in a local minimum.

### 2.2. Two stage evolutionary approach (Li et al., 2004)

The evolutionary approach consists of two major steps: (i) the first step aims to seek promising mathematical functions using genetic programming (GP). (ii) The second step is aimed at fitting parameters in the function formula thus found, using evolutionary programming (EP) techniques.

First we introduce GP (Koza and Andre, 1999) used in the modelling. In the case of GP here, we take a function set,  $F = \{+, -, *, /, \exp, \log, \sin, \cos\}$  and a terminal set,  $T = \{r, R\}$  where  $r$  is the variable radius and  $R$  is a random float-point value between  $-10.0$  and  $10.0$ . Theoretically, all profile models (1)–(3) could be potentially generated by GP.

The fitness function determines how well an individual expression fits an observational galaxy profile. The overall fitness function used here for GP consists of two items. We take hits as the first part of the fitness function. The hits measure here counts the number of fitness data points for which the numerical value returned by the model expression lies within a small tolerance (the hits criterion) of the observed intensity value. For example, the hits criterion taken in this paper is 0.005 for GP. Selecting an appropriate value for the hit criterion seems a bit tricky here. The aim of using GP is to find promising functional forms that potentially have capability for describing as many profiles as possible. Therefore, in our approach, taking a higher hit criterion for GP is preferable as long as the maximum number of hits (i.e., 50 in this study) is achievable in terms of the results on 18 galaxies profiles.

In addition, we add a second item into the fitness function to penalize any complex mathematical expression because a simple functional form is almost always preferable. The complexity of an expression is measured by the length of the expression, i.e., the number of nodes within the expression. The extent to which an expression should be penalized is changeable by a weight  $w$ . In summary, the fitness function that we use is given by

$$f = \text{Hits} - w * \text{the length of the expression.} \quad (5)$$

The criteria for terminating a GP run are either the maximum run time allowed or the maximum of generation that we set, whichever reached first.

In the second step, EP is used to tune parameters in the form. In this study, the EP algorithm is similar to the one used in Yao et al. (1999), except that self-adaptive mutation is not used. Cauchy mutation is used in the EP. The fitness function is defined by the hits, the first item of the fitness function for GP. The termination criterion taken here is that the maximum generation has been reached.

### 2.3. Experiments and results (Li et al., 2004)

The data are a set of 18 elliptical galaxies in the Coma cluster observed in near-infrared band (central wavelength 2.2  $\mu\text{m}$ ; for further description of the observations see Khosroshahi et al., 2000). They are also chosen from the same cluster of galaxies to enable us to eliminate the signature of different environ-

ments on the surface brightness of the sample galaxies. Elliptical galaxies historically have been characterized by the de Vaucouleurs ( $r^{1/4}$ ) profile, which is an empirical fit, not yet shown to emerge from theoretical or semi-analytic scenarios of galaxy formation. Careful observations reveal a far greater variety of profiles, leading, among others, Khosroshahi et al. (2000) to advocate the use of generalized models that allow for shallower or steeper profiles to elliptical galaxies and bulges of disk galaxies.

One-dimensional radial profiles have been extracted from fitting elliptical contours of the same brightness (isophotes) to the images of the galaxies, and computing the average brightness as a function of radius along the major axis of the ellipse at various values of the radial distance  $r$ . Each profile has 50 data points. The sky background is highly variable at near-infrared wavelengths. Therefore the local background is highly variable.

The experiments were carried out as follows. We first ran the GP system on each of 18 galaxy profile data sets. The details of parameter settings for running GP are given in the paper (Li et al., 2004).

One profile model is created after each GP run. 18 different profile models with different mathematical forms are generated in total. They seem different in their shapes of tree structure. Therefore, finding similarity between those forms generated is needed in order to find a more generic model form.

We simplify a tree-structured function form in the way of cutting each possible sub-tree and replacing it by one single numerical node if the ultimate return value of the sub-tree is a numerical value. After the simplification, the form has to be generalized in the way of replacing a few numerical values by variables (e.g.,  $a$ ,  $b$ , and  $c$ ), respectively.

After the generalization process, there emerge a few forms with parameters that are relatively simple. Among these forms we only select the following two simplest function forms  $f_{g1}$  and  $f_{g2}$  for further investigation.

$$f_{g1} = a + b/(c + r), \quad (6)$$

$$f_{g2} = a' + b'/(b' + c' r^2). \quad (7)$$

Each of the two forms involves three parameters i.e.,  $a$ ,  $b$ , and  $c$  for  $f_{g1}$ ;  $a'$ ,  $b'$  and  $c'$  for  $f_{g2}$ , and one radius  $r$ . Now, we use EP to seek proper values for three parameters within the both functions respectively to fit each galaxy profile in the logarithmic regime. The parameter settings for running EP are given in the paper (Li et al., 2004).

For brevity, results of fitting parameters found by EP for  $f_{g1}$  and for  $f_{g2}$  are both listed in Table 1.

Several points are worth indicating here. First of all, EP algorithms almost always achieve the maximum hits: 50 (the number of data points for each profile), based on either function forms for each galaxy profile. This is partly attributed to a large population of solutions. Secondly, unlike non-linear fitting algorithms, EP is not sensitive to the initial values provided at all. Throughout our experiments we keep the initial values for those parameters (i.e.,  $a$ ,  $b$ , and  $c$ ;  $a'$ ,  $b'$  and  $c'$ ), which do not need to be tuned in order to avoid failure of a run. This is attributed to the mutation in EP.

**Table 1 – Fitting parameters found by EP for functions  $f_{g1}$  and  $f_{g2}$  using 0.005 as a hit criterion**

| Profile | $f_{g2}$ |        |        | $f_{g2}$ |        |        |
|---------|----------|--------|--------|----------|--------|--------|
|         | a        | b      | c      | a'       | b'     | c'     |
| 1       | 3.7303   | 0.4272 | 1.7655 | 3.7681   | 0.1427 | 0.2193 |
| 2       | 3.7444   | 0.7167 | 4.8130 | 3.7886   | 0.0999 | 0.1485 |
| 3       | 3.7308   | 1.1017 | 6.0725 | 3.7700   | 0.1349 | 0.0943 |
| 4       | 3.6701   | 1.4376 | 6.2036 | 3.7276   | 0.1579 | 0.0386 |
| 5       | 3.6741   | 0.6300 | 3.0043 | 3.7073   | 0.1222 | 0.1078 |
| 6       | 3.6897   | 0.6138 | 2.1494 | 3.7272   | 0.1712 | 0.0872 |
| 7       | 3.6878   | 0.7763 | 4.8547 | 3.7316   | 0.1125 | 0.0663 |
| 8       | 3.7217   | 0.4731 | 2.3636 | 3.7547   | 0.1259 | 0.1589 |
| 9       | 3.7074   | 0.6060 | 2.4258 | 3.7347   | 0.1908 | 0.1721 |
| 10      | 3.6695   | 0.8550 | 2.5385 | 3.7093   | 0.2387 | 0.1462 |
| 11      | 3.6893   | 0.7154 | 5.7652 | 3.7253   | 0.0910 | 0.0884 |
| 12      | 3.6373   | 0.8105 | 3.7815 | 3.6796   | 0.1565 | 0.1438 |
| 13      | 3.6382   | 0.6320 | 2.7736 | 3.6808   | 0.1408 | 0.1257 |
| 14      | 3.6987   | 0.2636 | 2.1155 | 3.7096   | 0.0959 | 0.1510 |
| 15      | 3.6712   | 0.6221 | 2.3377 | 3.7002   | 0.1973 | 0.1582 |
| 16      | 3.6804   | 0.8437 | 2.9601 | 3.6999   | 0.2267 | 0.0779 |
| 17      | 3.6960   | 1.4928 | 4.4057 | 3.7479   | 0.2550 | 0.0623 |
| 18      | 3.6515   | 1.4413 | 4.7698 | 3.7065   | 0.2213 | 0.0741 |

Thirdly, the hit criterion is one of the most important parameter settings in EP because it determines how well the resulting models fit the empirical galaxy profile. The smaller the value of the hit criterion is, the better fit the resulting model could be.

To evaluate the accuracy and overall success of a model fit, we use statistical measure, the reduced  $\chi^2$ , given in Eq. (4). In general, the resultant  $\chi^2$  above 2 means that the model does not seem to describe the empirical galaxy profile very well. Otherwise, the model is a good description for the brightness distribution of a galaxy. Results are shown under the columns with hit criterion=0.005 in Table 2.

According to the value of the reduced  $\chi^2$ , only one model based on  $f_{g1}$  fits the observational profile very well (i.e., galaxy 18; its  $\chi^2=1.2161<2$ ) whereas three models based on  $f_{g2}$  perform well for galaxy 10, galaxy 17 and galaxy 18 as each  $\chi^2$  is less than 2. We may argue that  $f_{g2}$  is a better model than  $f_{g1}$  in

terms of the total number of good model fittings. Apart from this fact, the results do not seem promising at all. The poor performance is partly due to the higher hit criterion we set in EP.

## 2.4. Discussions

There are many modelling problems in ecoinformatics, some of which can benefit a lot from the two stage evolutionary approach described in previous sections. If there is very little domain knowledge available for a given modelling problem and the only information we have is just a data set, then GP would be a very good choice in finding a human comprehensible model, which could then be fine-tuned later. If there is rich domain knowledge, it can be used to constrain the search space and make evolutionary search more efficient. If the domain knowledge is sufficiently rich to determine the structure of a model, evolutionary algorithms (e.g., EP) can be used to calibrate the model and optimise the model parameters.

## 3. Coevolutionary problem solving

Coevolution is an idea that computer scientists borrowed from ecologists. It has been used widely in evolutionary computation to solve large and complex problems. For example, almost all evolutionary algorithms in the literature were tested on problems whose dimensions (i.e., the number of real-valued variables, not binary bits) vary from dozens to a couple of hundreds. Few were tackling problems with the number of variables up to a thousand. Coevolution could be harnessed to deal with problems of a very large scale.

### 3.1. Fast evolutionary programming with cooperative coevolution (Liu et al., 2001)

Fast evolutionary programming (FEP) algorithm is based on Cauchy mutation and has excellent performance on

**Table 2 – The reduced  $\chi^2$  for 18 galaxies models based on two function  $f_{g1}$  and  $f_{g2}$  using two hit criteria**

| Profiles | $\chi^2$ for $f_{g1}$    |                          | $\chi^2$ for $f_{g2}$    |                          |
|----------|--------------------------|--------------------------|--------------------------|--------------------------|
|          | Using hit criteria=0.005 | Using hit criteria=0.002 | Using hit criteria=0.005 | Using hit criteria=0.002 |
| 1        |                          | 7.4460                   | 11.4012                  | 1.1141                   |
| 2        |                          | 9.1809                   | 12.8405                  | 1.8633                   |
| 3        |                          | 8.7796                   | 6.3452                   | 0.7113                   |
| 4        |                          | 6.8782                   | 2.8035                   | 2.8385                   |
| 5        |                          | 4.1948                   | 5.7141                   | 0.8685                   |
| 6        |                          | 3.1867                   | 3.2442                   | 0.7194                   |
| 7        |                          | 9.7797                   | 17.9798                  | 0.9289                   |
| 8        |                          | 6.8756                   | 11.0505                  | 0.1353                   |
| 9        |                          | 5.8373                   | 3.4484                   | 1.0390                   |
| 10       |                          | 2.8263                   | 1.9078                   | 0.3318                   |
| 11       |                          | 34.0802                  | 25.2975                  | 7.9733                   |
| 12       |                          | 5.8477                   | 7.9886                   | 1.5150                   |
| 13       |                          | 7.6026                   | 10.6681                  | 1.2615                   |
| 14       |                          | 9.1534                   | 13.8083                  | 1.2896                   |
| 15       |                          | 5.9548                   | 2.1233                   | 1.2062                   |
| 16       |                          | 4.8048                   | 3.7705                   | 0.6759                   |
| 17       |                          | 2.9778                   | 1.3418                   | 0.3138                   |
| 18       |                          | 1.2161                   | 0.8031                   | 0.0796                   |

difficult multi-modal functions (Yao et al., 1999). However, the problems studied in Yao et al. (1999) have only 30 dimensions, which are relatively small for many real-world problems. It is interesting to know whether the results obtained from the low dimensional problems can be generalised to higher-dimensional problems. It is also important to investigate how FEP scales as the problems size increases. Unfortunately, the reported studies on the scalability of evolutionary algorithms (including FEP) are scarce. It was discovered previously that neither classical EP (CEP) nor FEP performed satisfactorily for some large-scale problems (Yao and Liu, 1998a,b). The reason for FEP’s poor performance on the high dimensional problem lies in its overly large search step size. The step size of the Cauchy mutation increases monotonically as the dimensionality increases. As pointed out in a previous study (Yao et al., 1999), there is an optimal search step size for a given problem. A step size which is too large or too small will have a negative impact on the performance of search.

To tackle large problems, FEP with cooperative coevolution (FEPCC) has been proposed (Liu et al., 2001). It adopts the divide-and-conquer strategy and divides the problem into many sub-problems, each of which is then solved by FEP. It repeats the following two steps for many generations until a good solution to the entire problem is found:

1. Evolve each module separately, and
2. Combine them to form the whole system.

In the following sections, we will describe in more detail FEPCC for large-scale problems with dimensions ranging from 100 to 1000. The problems used in our empirical studies include four unimodal functions and four multimodal functions with many local optima.

### 3.1.1. Fast evolutionary programming

The FEP applied to function optimisation can be described as follows (Yao et al., 1999):

1. Generate the initial population of  $\mu$  individuals, and set  $k=1$ . Each individual is taken as a pair of real-valued vectors,  $(\mathbf{x}_i, \eta_i)$ ,  $\forall i \in \{1, \dots, \mu\}$  where  $\mathbf{x}_i$ ’s are objective variables and  $\eta_i$ ’s are standard deviations for Cauchy mutations (also known as strategy parameters in self-adaptive evolutionary algorithms).
2. Evaluate the fitness score for each individual  $(\mathbf{x}_i, \eta_i)$ ,  $\forall i \in \{1, \dots, \mu\}$ , of the population based on the objective function,  $f(\mathbf{x}_i)$ .
3. Each parent  $(\mathbf{x}_i, \eta_i)$ ,  $i=1, \dots, \mu$ , creates a single offspring  $(\mathbf{x}'_i, \eta'_i)$  by: for  $j=1, \dots, n$ ,

$$\eta'_i(j) = \eta_i(j) \exp(\tau N(0,1) + \tau' N_j(0,1)), \quad (8)$$

$$\mathbf{x}'_i(j) = \mathbf{x}_i(j) + \eta_i(j) \delta_j, \quad (9)$$

where  $\eta_i(j)$ ,  $\eta'_i(j)$ ,  $\mathbf{x}_i(j)$ , and  $\mathbf{x}'_i(j)$  denote the  $j$ th component of the vectors  $\eta_i$ ,  $\eta'_i$ ,  $\mathbf{x}_i$  and  $\mathbf{x}'_i$ , respectively. The factors  $\tau$  and  $\tau'$  are commonly set to  $(\sqrt{2\sqrt{n}})^{-1}$  and  $(\sqrt{2n})^{-1}$  (Yao et al., 1999).  $N(0, 1)$  denotes a normally distributed one-dimensional random number with mean 0 and standard deviation 1.  $N_j(0, 1)$  indi-

cates that the random number is generated anew for each value of  $j$ .  $\delta_j$  is a Cauchy random variable with the scale parameter  $t=1$ , and is generated anew for each value of  $j$ . The one-dimensional Cauchy density function centered at the origin is defined by:

$$f_t(x) = \frac{1}{\pi} \frac{t}{t^2 + x^2}, \quad -\infty < x < \infty, \quad (10)$$

where  $t>0$  is a scale parameter (Feller, 1971) (pp. 51). The shape of  $f_t(x)$  resembles that of the Gaussian density function but approaches the axis so slowly that an expectation does not exist. As a result, the variance of the Cauchy distribution is infinite.

4. Calculate the fitness of each offspring  $(\mathbf{x}'_i, \eta'_i)$ ,  $\forall i \in \{1, \dots, \mu\}$ .
5. Conduct pairwise comparison over the union of parents  $(\mathbf{x}_i, \eta_i)$  and offspring  $(\mathbf{x}'_i, \eta'_i)$ ,  $\forall i \in \{1, \dots, \mu\}$ . For each individual,  $q$  opponents are chosen uniformly at random from all the parents and offspring. For each comparison, if the individual’s fitness is no smaller than the opponent’s, it receives a “win”.
6. Select the  $\mu$  individuals out of  $(\mathbf{x}_i, \eta_i)$  and  $(\mathbf{x}'_i, \eta'_i)$ ,  $\forall i \in \{1, \dots, \mu\}$ , that have the most wins to be parents of the next generation.
7. Stop if the halting criterion is satisfied; otherwise,  $k=k+1$  and go to Step 3.

### 3.1.2. Cooperative coevolution (Liu et al., 2001)

Given that a solution to a function minimisation problem consists of a vector of  $n$  variable values, a direct cooperative coevolution for function optimisation is to assign a population to each variable, and coevolve the individuals in each separate population. One complete evolution of all populations for  $n$  variables during one generation is called a cycle. A cycle of evolution of FEPCC consists of three major steps:

1.  $j=1$ . Apply FEP to the population of variable 1.
2.  $j=j+1$ . Apply FEP to the population of variable  $j$ .
3. If  $j \leq n$ , go to Step 2. Otherwise, go to the next cycle.

To reduce the evaluation time, the fitness of an individual in FEPCC was estimated by combining it with the current best individuals from each of the other populations to form a vector of real values, and applying the vector to the target function. The fitness evaluation in FEPCC is different with that in FEP. In FEP, the target function value of a new individual have to be recalculated since an individual is a vector and the most of components in the vector have new values. In contrast, because an individual in FEPCC is a component in a vector and other components in the vector remain unchanged, it only needs to calculate the difference caused by the changed component. This fitness evaluation method used in this paper takes less computation time.

### 3.2. Experimental studies (Liu et al., 2001)

#### 3.2.1. Benchmark functions

The eight benchmark functions used here are given in Table 3. Functions  $f_1$  to  $f_4$  are unimodal. Functions  $f_5$  to  $f_8$  are

**Table 3 – The eight benchmark functions used in our experimental study, where  $n$  is the dimension of the function,  $f_{\min}$  is the minimum value of the function, and  $S \subseteq \mathbb{R}^n$** 

| Test function   | $S$               | $f_{\min}$   |
|---|-------------------|--------------|
| $f_1(x) = \sum_{i=1}^n x_i^2$   | $[-100, 100]^n$   | 0            |
| $f_2(x) = \sum_{i=1}^n  x_i  + \prod_{i=1}^n  x_i $   | $[-10, 10]^n$     | 0            |
| $f_3(x) = \max_i \{ x_i , 1 \leq i \leq n\}$  | $[-100, 100]^n$   | 0            |
| $f_4(x) = \sum_{i=1}^n (ix_i + 0.5)^2$  | $[-100, 100]^n$   | 0            |
| $f_5(x) = \sum_{i=1}^n -x_i \sin(\sqrt{ x_i })$   | $[-500, 500]^n$   | $-418.9829n$ |
| $f_6(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$  | $[-5.12, 5.12]^n$ | 0            |
| $f_7(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos 2\pi x_i\right) + 20 + e$ | $[-32, 32]^n$     | 0            |
| $f_8(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$  | $[-600, 600]^n$   | 0            |

multimodal functions where the number of local minima increases exponentially with the problem dimension (Törn and Žilinskas, 1989; Schwefel, 1995).

In the experiments, the population size  $\mu=50$ , the tournament size  $q=10$  for selection, and the initial  $\mu=3.0$  were used. The initial population was generated uniformly at random in the range as specified in Table 3.

### 3.2.2. Results on unimodal functions

The first set of experiments was aimed to study the convergence rate of FEPCC for functions  $f_1$  to  $f_4$  with different dimensions. The dimensions of  $f_1$ – $f_4$  were set to 100, 250, 500, 750, and 1000, respectively, in our experiments. For a function in each dimension, 50 independent runs were conducted. The average results of 50 runs are summarised in Table 4. Fig. 1 shows the progress of the mean solutions found over 50 runs for  $f_1$  to  $f_4$ . Function  $f_1$  is the simple sphere model studied by many researchers. Function  $f_3$  is a

discontinuous function. FEPCC maintains a nearly constant convergence rate throughout the evolution for functions  $f_1$  and  $f_3$ . For function  $f_2$ , when the term of  $\prod_{i=1}^n |x_i|$  dominates the target function value, FEPCC displays a fast convergence rate. Once the term of  $\sum_{i=1}^n |x_i|$  takes over, FEPCC's convergence rate reduces substantially. Function  $f_4$  is the step function that is characterized by plateaus and discontinuity. FEPCC moves quickly from one plateau to a lower one for  $f_4$ .

According to Fig. 1 and Table 4, the computational time used by FEPCC to find an optimal solution to the unimodal functions appears to grow in the order of  $O(n)$ . For example, for function  $f_1$ , 500,000 fitness evaluations were need for  $n=100$ , 1,250,000 for  $n=250$ , 2,500,000 for  $n=500$ , 3,750,000 for  $n=750$ , and 5,000,000 for  $n=1000$ . Although the number of fitness evaluation is relatively large in FEPCC, the total computation time is considerably short. An individual in a population in FEPCC is a single variable. Its fitness can be easily evaluated from its parent. Only the difference caused by one changed variable in a vector needs to be calculated.

**Table 4 – The mean solutions found for  $f_1$ – $f_4$ , with dimension  $n=100, 250, 500, 750$ , and 1000, respectively, by FEPCC**

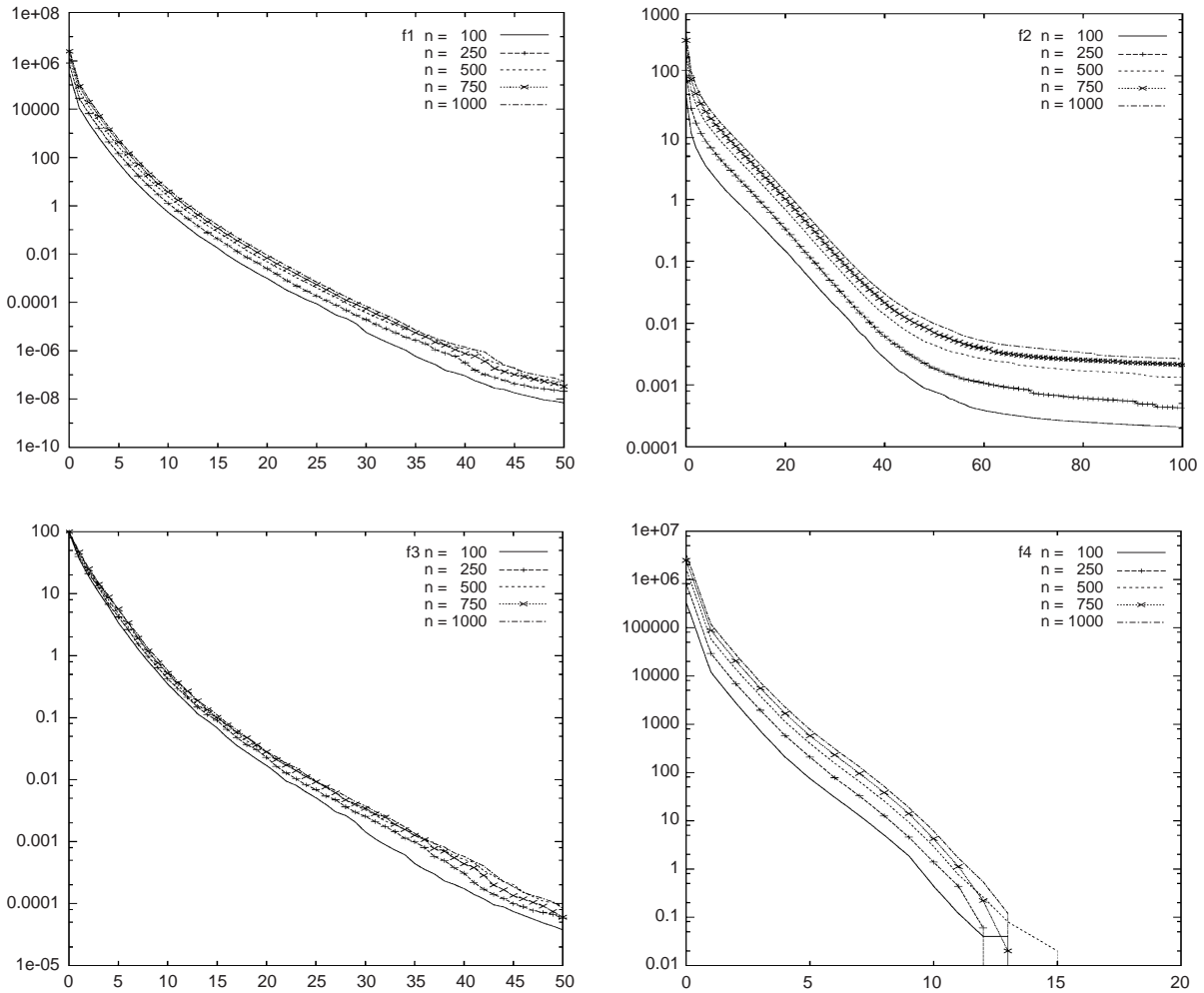
| Function | $n$  | No. of cycle | Mean                 | Standard deviation   |
|----------|------|--------------|----------------------|----------------------|
| $f_1$    | 100  | 50           | $6.8 \times 10^{-9}$ | $1.8 \times 10^{-8}$ |
|          | 250  | 50           | $2.1 \times 10^{-8}$ | $6.2 \times 10^{-7}$ |
|          | 500  | 50           | $4.9 \times 10^{-8}$ | $1.2 \times 10^{-7}$ |
|          | 750  | 50           | $3.4 \times 10^{-8}$ | $1.1 \times 10^{-8}$ |
|          | 1000 | 50           | $5.4 \times 10^{-8}$ | $2.8 \times 10^{-8}$ |
| $f_2$    | 100  | 100          | $2.1 \times 10^{-4}$ | $6.1 \times 10^{-4}$ |
|          | 250  | 100          | $4.3 \times 10^{-4}$ | $7.4 \times 10^{-4}$ |
|          | 500  | 100          | $1.3 \times 10^{-3}$ | $3.0 \times 10^{-3}$ |
|          | 750  | 100          | $2.1 \times 10^{-3}$ | $2.6 \times 10^{-3}$ |
|          | 1000 | 100          | $2.6 \times 10^{-3}$ | $3.2 \times 10^{-3}$ |
| $f_3$    | 100  | 50           | $3.8 \times 10^{-5}$ | $4.8 \times 10^{-5}$ |
|          | 250  | 50           | $5.8 \times 10^{-5}$ | $9.1 \times 10^{-5}$ |
|          | 500  | 50           | $9.0 \times 10^{-5}$ | $1.5 \times 10^{-4}$ |
|          | 750  | 50           | $6.0 \times 10^{-5}$ | $3.0 \times 10^{-5}$ |
|          | 1000 | 50           | $8.5 \times 10^{-5}$ | $7.2 \times 10^{-5}$ |
| $f_4$    | 100  | 50           | 0.0                  | 0.0                  |
|          | 250  | 50           | 0.0                  | 0.0                  |
|          | 500  | 50           | 0.0                  | 0.0                  |
|          | 750  | 50           | 0.0                  | 0.0                  |
|          | 1000 | 50           | 0.0                  | 0.0                  |

One complete evolution of all populations in one generation is called a cycle. The function evaluation number in one cycle equals  $100n$ .

### 3.2.3. Results on multimodal functions

Multimodal functions having many local minima are often regarded as being difficult to optimise.  $f_5$ – $f_8$  are such functions where the number of local minima increases exponentially as the dimension of the function increases. Five different values of  $n$  have been used in our experiments, i.e.,  $n=100, 250, 500, 750$ , and 1000. The average results of 50 runs are summarised in Table 5. Fig. 2 shows the progress of the mean solutions found over 50 runs for  $f_5$  to  $f_8$ . According to Fig. 2, FEPCC was able to improve its solution steadily for a long time for functions  $f_6, f_7$ , and  $f_8$ , but fell to a local optimum quite early for function  $f_5$ . One reason for FEPCC becoming trapped in a local optimum is that it used a greedy fitness evaluation. The fitness of an individual is evaluated based on the vector formed by this individual and the current best individuals from each of the other populations. To get a better evaluation, we can construct many vectors, and determine the fitness of an individual by evaluating the target function values of all vectors containing this individual.

Based on Fig. 2 and Table 5, the computational time used by FEPCC to find a near optimal solution to the multimodal functions also appears to grow in the order of  $O(n)$ . For example, for



**Fig. 1**–The evolution process of the mean best values found for  $f_1$ – $f_4$ , with dimension  $n=100, 250, 500, 750,$  and  $1000,$  respectively, by FEPC. The results were averaged over 50 runs. The vertical axis is the function value and the horizontal axis is the number of cycles. The function evaluation number in one cycle equals  $100n$ .

function  $f_8$ , 500,000 fitness evaluations were need for  $n=100,$  1,250,000 for  $n=250,$  2,500,000 for  $n=500,$  3,750,000 for  $n=750,$  and 5,000,000 for  $n=1000.$

**3.2.4. Comparisons with fast evolutionary programming.** Tables 6 and 7, and Figs. 3 and 4 summarise the average results of FEPC in comparison with FEP on the sphere function  $f_1$  and the Ackley’s function  $f_8$ . Only the results of FEP on  $f_1$  and  $f_8$ , with  $n=100, 200,$  and  $300,$  respectively, were reported in (Yao and Liu, 1998a). When the dimensionality becomes large, FEP converges very slowly in comparison with FEPC. The reason that FEP’s performance worsens as the dimensionality increases lies in its increasingly large search step sizes. FEP’s search step size (driven by the search step size of the Cauchy mutation) increases as the dimensionality increases. When the search step size is larger than the optimal one, further increase in the step size can only deteriorate the search performance (Yao et al., 1999). FEPC is applied to a variable of a vector rather than the whole vector, the problem of too large step size with high dimensionality does not exist. Furthermore, the robust and faster search capability of Cauchy mutation in one di-

mension can be fully explored by cooperative coevolution approach.

**3.3. Discussion**

It is shown through FEPC that coevolution can be used quite simply for enhancing the performance of existing algorithms. The time used by FEPC to find a near optimal solution appears to increase linearly as the dimensionality increases. FEPC represents only one of the possibilities that coevolution can be utilised. There are other uses of coevolution which can help to tackle large and complex problems. Ecoinformatics appears to be an excellent field where coevolution can play a major role.

**4. Automatic problem decomposition and neural network ensembles**

Many real-world problems are too large and too complex for a single monolithic system to solve alone. There are many examples from both natural and artificial systems

**Table 5 – The mean solutions found for  $f_5$ – $f_8$ , with dimension  $n=100, 250, 500, 750$ , and  $1000$ , respectively, by FEPCC**

| Function | $n$  | No. of cycle | Mean                 | Standard deviation   |
|----------|------|--------------|----------------------|----------------------|
| $f_5$    | 100  | 50           | –1,867.3             | 52.28                |
|          | 250  | 50           | –104,677.2           | 96.47                |
|          | 500  | 50           | –209,316.4           | 121.3                |
|          | 750  | 50           | –313,995.8           | 171.1                |
|          | 1000 | 50           | –18,622.6            | 200.6                |
| $f_6$    | 100  | 50           | 0.026                | 0.14                 |
|          | 250  | 50           | 0.048                | 0.15                 |
|          | 500  | 50           | 0.143                | 0.28                 |
|          | 750  | 50           | 0.163                | 0.23                 |
|          | 1000 | 50           | 0.313                | 0.40                 |
| $f_7$    | 100  | 50           | $1.7 \times 10^{-4}$ | $2.1 \times 10^{-5}$ |
|          | 250  | 50           | $3.5 \times 10^{-4}$ | $3.0 \times 10^{-5}$ |
|          | 500  | 50           | $5.7 \times 10^{-4}$ | $3.9 \times 10^{-5}$ |
|          | 750  | 50           | $7.8 \times 10^{-4}$ | $4.1 \times 10^{-5}$ |
|          | 1000 | 50           | $9.5 \times 10^{-4}$ | $3.4 \times 10^{-5}$ |
| $f_8$    | 100  | 50           | 0.047                | 0.065                |
|          | 250  | 50           | 0.025                | 0.052                |
|          | 500  | 50           | 0.029                | 0.085                |
|          | 750  | 50           | 0.061                | 0.195                |
|          | 1000 | 50           | 0.025                | 0.114                |

One complete evolution of all populations in one generation is called a cycle. The function evaluation number in one cycle equals  $100n$ .

that show that an integrated system consisting of several subsystems can reduce the total complexity of the system while solving a difficult problem satisfactorily. Neural network (NN) ensembles is a typical example (Yao and Liu, 1998b).

NN ensembles adopt the divide-and-conquer strategy. Instead of using a single network to solve a task, an NN ensemble combines a set of NNs that learn to subdivide the task and thereby solve it more efficiently and elegantly. An NN ensemble offers several advantages over a monolithic NN. First, it can perform more complex tasks than any of its components (i.e., individual NNs in the ensemble). Second, it can make an overall system easier to understand and modify. Finally, it is more robust than a monolithic NN, and can show graceful performance degradation in situations where only a subset of NNs in the ensemble are performing correctly.

#### 4.1. Negative correlation learning (Liu and Yao, 1999a)

The negative correlation learning has been successfully applied to NN ensembles (Liu and Yao, 1999b,c; Liu, Yao and Higuchi, 2000; Monirul Islam et al., 2003) for creating negatively correlated NNs (NCNNs). It introduces a correlation penalty term into the error function of each individual NN in an ensemble so that the individual NN can be trained cooperatively. Specially, the error function  $E_i$  for individual  $i$  is defined by

$$E_i = \frac{1}{N} \sum_{n=1}^N E_i(n) = \frac{1}{N} \sum_{n=1}^N \left[ \frac{1}{2} (d(n) - F_i(n))^2 + \lambda p_i(n) \right], \quad (11)$$

where  $N$  is the number of training patterns,  $E_i(n)$  is the value of the error of individual NN  $i$  at presentation of the

$n$ th training pattern,  $d(n)$  refers the desired response for the  $n$ th training pattern,  $F_i(n)$  is the output of individual NN  $i$  on the  $n$ th training pattern, and  $p_i$  is a correlation penalty function. The purpose of minimising  $p_i$  is to negatively correlate each individual's error with errors for the rest of the ensemble. The parameter  $\lambda > 0$  is used to adjust the strength of the penalty. The function  $p_i$  can be chosen as

$$p_i(n) = (F_i(n) - d(n)) \sum_{j \neq i} (F_j(n) - d(n)). \quad (12)$$

The partial derivative of  $E_i$  with respect to the output of individual  $i$  on  $n$ th training pattern is

$$\frac{\partial E_i(n)}{\partial F_i(n)} = F_i(n) - d(n) + \lambda \sum_{j \neq i} (F_j(n) - d(n)). \quad (13)$$

The standard back-propagation (BP) algorithm with pattern-by-pattern updating (Haykin, 1994) can be used for weight adjustments in our ensembles. Weight updating of NNs is performed using Eq. (13) after the presentation of each training pattern. One complete presentation of the entire training set during the learning process is called an epoch.

The sum of  $E_i(n)$  over all  $i$  is

$$\begin{aligned} E(n) &= \sum_{i=1}^M E_i(n) \\ &= \left( \frac{1}{2} - \lambda \right) \sum_{i=1}^M (d(n) - F_i(n))^2 + \lambda \left( \sum_{i=1}^M F_i(n) - Md(n) \right)^2. \end{aligned} \quad (14)$$

From Eqs. (13) and (14), the following observations can be made:

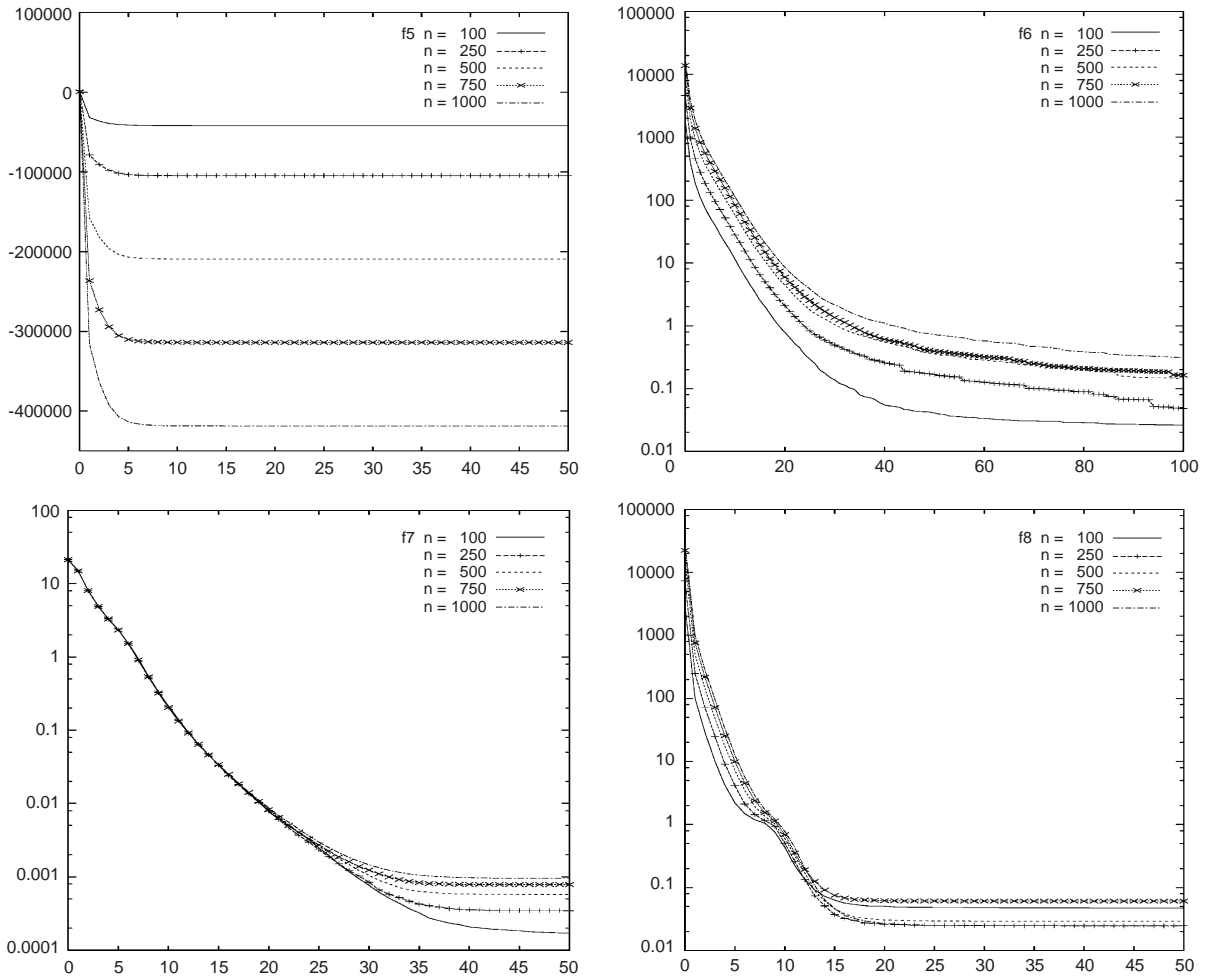
1. During the training process, all the individual networks interact with each other through their penalty terms in the error functions.
2. For  $\lambda=0.0$ , there are no correlation penalty terms in the error functions of the individual networks, and the individual networks are trained independently using BP. That is, independent training using BP for the individual networks is a special case of our negative correlation learning.
3. For  $\lambda=1/2$ , Eq. (14) can be rewritten as

$$E(n) = \frac{1}{2} \left( \sum_{i=1}^M F_i(n) - Md(n) \right)^2. \quad (15)$$

The right term in Eq. (15), denoted by  $E_{ave}$ , is the error function of the ensemble. From this point of view, negative correlation learning provides a novel way to decompose the learning task of the ensemble into a number of subtasks for each individual.

4. For  $\lambda=1$ , the following equality holds

$$\frac{\partial E_{ave}}{\partial F_i(n)} = \frac{\partial E_i(n)}{\partial F_i(n)}. \quad (16)$$



**Fig. 2**–The evolution process of the mean best values found for  $f_5$ – $f_8$ , with dimension  $n=100, 250, 500, 750$ , and  $1000$ , respectively, by FEPCC. The results were averaged over 50 runs. The vertical axis is the function value and the horizontal axis is the number of cycles. The function evaluation number in one cycle equals  $100n$ .

The minimisation of the error function of the ensemble is achieved by minimising the error functions of the individual networks.

**4.2. Chlorophyll-a prediction (Liu and Yao, 1999a)**

Lake Kasumigaura is situated in the South-Eastern part of Japan. It has a large and shallow water body where no

thermal stratification occurs. The annual water temperature of the lake ranges from 4 °C to 30 °C in summer. Due to high external and internal nutrient loadings, the primary productivity of the lake is extremely high, and favours harmful blue-green algae such as *Microcystis* and *Oscillatoria*. As algal succession changes species abundance year by year, it is very difficult to causally determine and predict algal blooms in Lake Kasumigaura (Recknagel, 1997).

**4.2.1. Experimental setup (Liu and Yao, 1999a)**

NCNNs were tested to predict timing and magnitudes for chlorophyll-a in Lake Kasumigaura by exploration of limnological time series for 10 years. Eight years of historical data were used for training and the data of two independent years 1986 and 1993 were chosen for testing. The year 1986 was chosen for testing as it was typical for blooms of *Microcystis* as observed in the years before 1987. The year 1993 was considered as typical for blooms by *Oscillatoria* as observed from 1987 afterwards (Recknagel, 1997).

The training data were divided into two parts, 1984–1985 and 1987–1992. This allowed to train two NN ensembles: one

**Table 6** – Comparison between FEPCC with FEP on function  $f_1$  (the sphere function) with  $n=100, 200$  and  $300$ , respectively

| n   | FEPCC                       |                      | FEP                         |                      |
|-----|-----------------------------|----------------------|-----------------------------|----------------------|
|     | No. of function evaluations | Mean of solutions    | No. of function evaluations | Mean of solutions    |
| 100 | $5.0 \times 10^5$           | $6.8 \times 10^{-9}$ | $7.5 \times 10^5$           | $4.7 \times 10^{-3}$ |
| 200 | $1.0 \times 10^6$           | $1.4 \times 10^{-8}$ | $1.5 \times 10^6$           | $9.1 \times 10^{-2}$ |
| 300 | $1.5 \times 10^6$           | $1.6 \times 10^{-8}$ | $3.0 \times 10^6$           | 0.46                 |

The results of FEPCC were averaged over 50 independent runs. The results of FEP were averaged over 10 independent runs.

**Table 7 – Comparison between FEPCC with FEP on function  $f_8$  (the Ackley’s function) with  $n=100, 200$  and  $300$ , respectively**

| n   | FEPCC                       |                      | FEP                         |                      |
|-----|-----------------------------|----------------------|-----------------------------|----------------------|
|     | No. of function evaluations | Mean of solutions    | No. of function evaluations | Mean of solutions    |
| 100 | $5.0 \times 10^5$           | $1.7 \times 10^{-4}$ | $7.5 \times 10^5$           | $3.7 \times 10^{-2}$ |
| 200 | $1.0 \times 10^6$           | $3.1 \times 10^{-4}$ | $1.5 \times 10^6$           | 15.2                 |
| 300 | $1.5 \times 10^6$           | $3.6 \times 10^{-4}$ | $3.0 \times 10^6$           | 20.7                 |

The results of FEPCC were averaged over 50 independent runs. The results of FEP were averaged over 10 independent runs.

with conditions that caused the dominance of Microcystis, and the other with conditions which favour Oscillatoria. The input for each individual network consists current input conditions and output conditions in the past 7 days. The values of training and testing data were rescaled linearly to between 0.1 and 0.9. The data are described in more detail in Recknagel (1997).

The normalised root-mean-square (RMS) error E was used to evaluate the performance of NCNNs (Liu and Yao, 1999a). The ensemble architectures used in the experiments have 6 individual networks. Each individual network is a feedforward

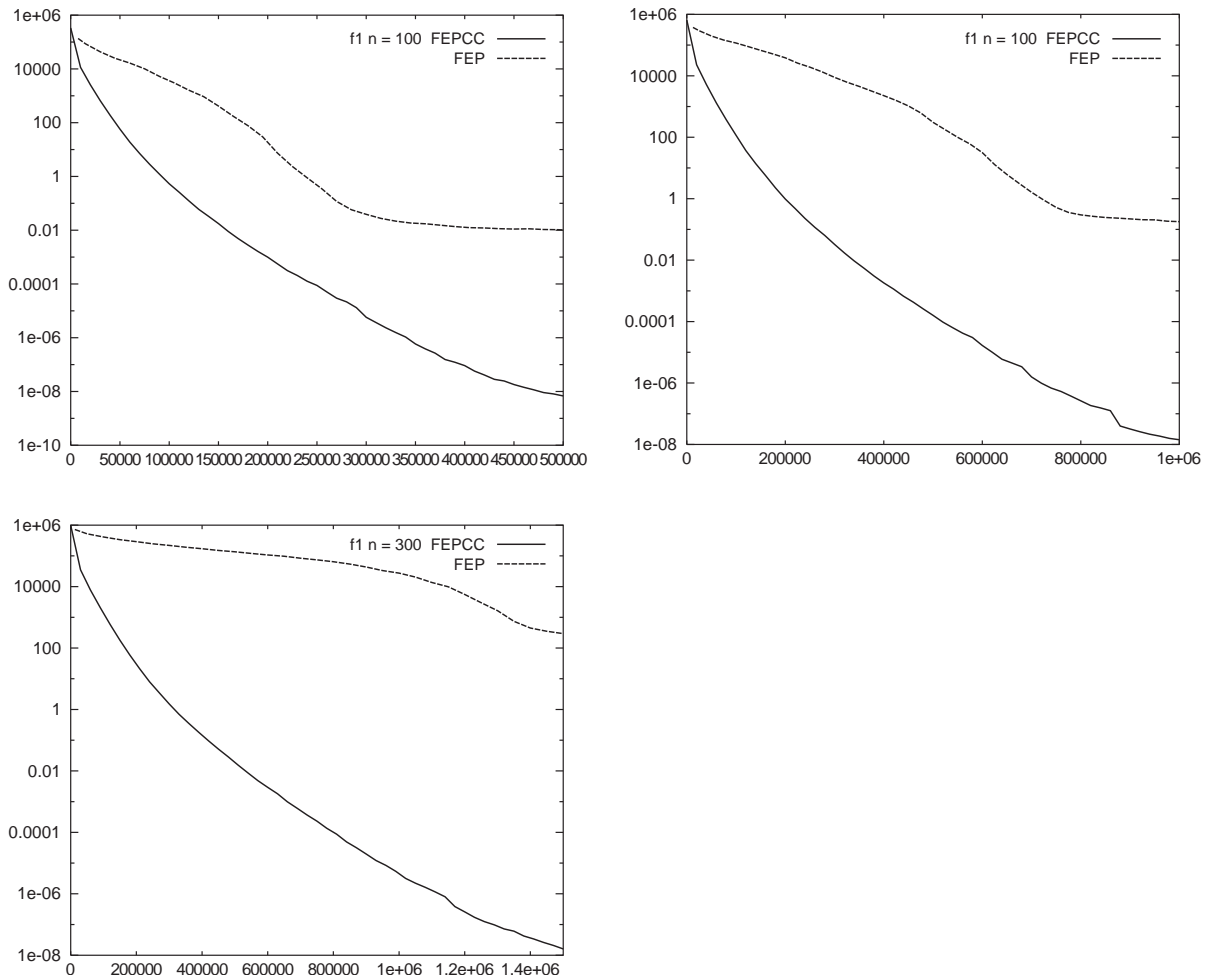
NN with one hidden layer. Both hidden node function and output node function are defined by the logistic function. All the individual networks have 15 hidden nodes. The number of training epochs was set to 2000 for the first part of training data, and 1000 for the second part of training data. The learning rate  $\eta$  and strength parameter  $\lambda$  were set to 0.25 and 1.0, respectively.

4.2.2. Experimental results (Liu and Yao, 1999a)

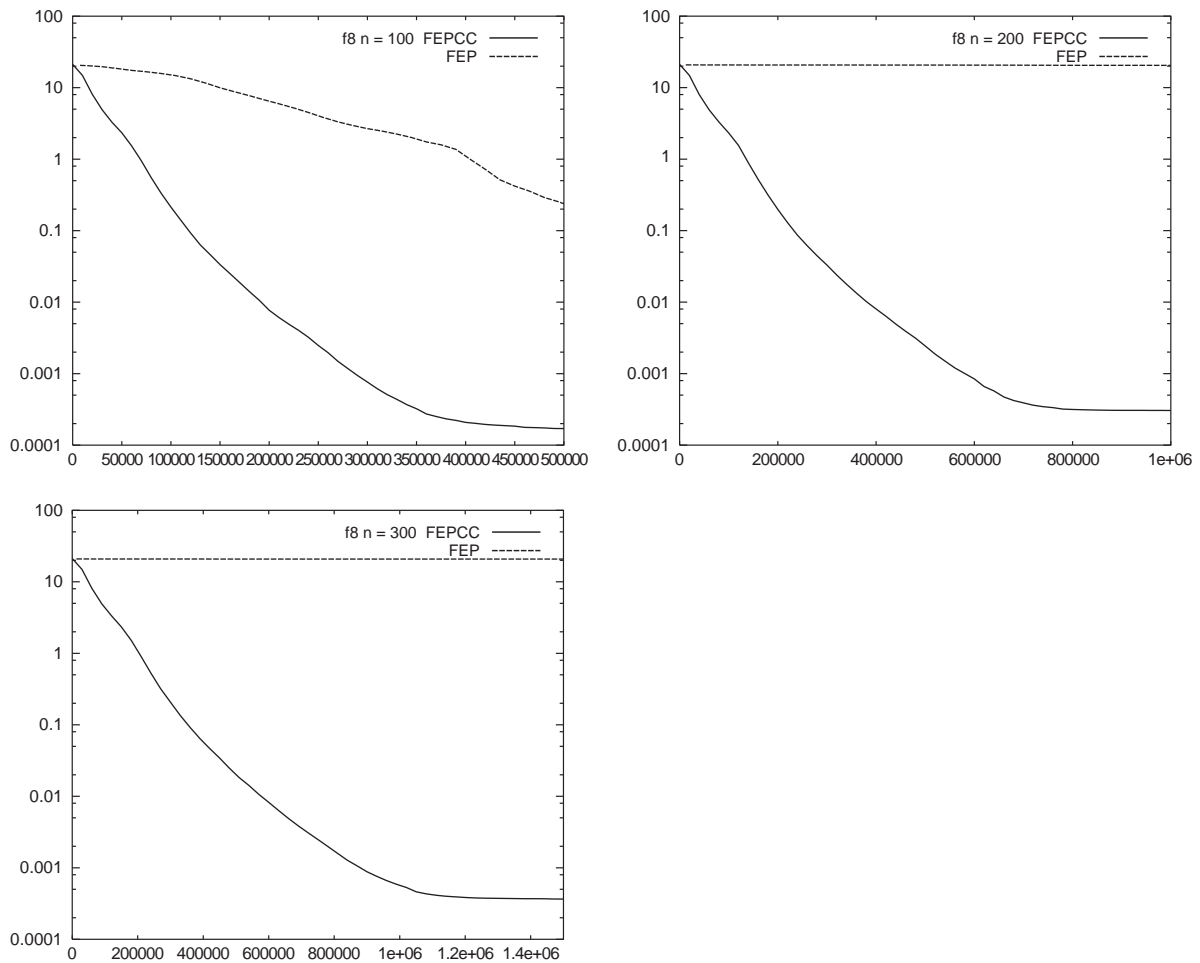
Table 8 shows the average results of NCNNs over 50 runs for the chlorophyll-a prediction problem. Each run of NCNNs was from different initial weights. Fig. 5 shows the best predictions of NCNNs along with the observed values. As can be seen, the predictions of NCNNs are remarkably accurate except the peak magnitudes in 1986 were slightly underestimated. NCNNs also outperformed the standard feedforward NNs for the chlorophyll-a prediction problem (Recknagel, 1997).

4.3. Discussion

As shown in the example of chlorophyll-a prediction, an ensemble of simple NNs can perform much better than a monolithic NN designed by human specialists. Although



**Fig. 3 – Comparison between FEPCC and FEP on  $f_1$  with dimension  $n=100, 200$ , and  $300$ , respectively. The results of FEPCC were averaged over 50 independent runs. The results of FEP were averaged over 10 independent runs.**



**Fig. 4– Comparison between FEPCC and FEP on  $f_8$  with dimension  $n=100, 200,$  and  $300,$  respectively. The results of FEPCC were averaged over 50 independent runs. The results of FEP were averaged over 10 independent runs.**

the ensemble structure (e.g., the number of NNs in the ensemble) was manually designed in the chlorophyll-a prediction example, there are automatic algorithms that can design the ensemble structure and individual NN structures (Liu et al., 2000; Monirul Islam et al., 2003). Negative correlation is essential in encouraging error diversity in the ensemble (Brown et al., 2005). Multi-objective evolutionary algorithms can be used quite nicely in designing and evolving NN ensembles that have good accuracy and diversity (Chandra and Yao, 2004, 2005; Chandra and Yao, accepted for publication).

## 5. Ecosystem-inspired computational models and algorithms

Evolutionary algorithms and neural networks are useful tools for modelling complex systems, e.g., biological ecosystems. There has been much work in the direction of applying latest computational tools to ecoinformatics problems. However, the work in the opposite direction, i.e., the development of new computational techniques and algorithms that inspired by biological ecosystems, has been underexplored and can be very rewarding for ecoinformatics researchers.

This section presents some possible directions for future research.

### 5.1. Robustness and diversity

As a problem becomes large and more complex, it is increasingly more difficult to find a monolithic problem-solving system to tackle it. For example, we often use ensembles or populations of many small neural networks, rather than a single monolithic neural network, to solve large and complex problems (Liu et al., 2000). Combining multiple classifiers is actually a very common practice in machine learning and pattern recognition. Similarly, we use populations of circuits, rather than a single one, to compose a reliable system (Schnier and Yao, 2003; Schnier et al., 2004). Even in optimisation, we use a population of different search operators (Yao et al., 1999; Lee and Yao, 2004). In all of these problem-solving systems, diversity is a crucial factor in influencing the performance of the system. If the diversity among different components of the system is high, the problem solving system is more likely to perform well for different problems in the presence of noise. It is also less likely to fail just because one or two components fail. Robustness of the system, in this

**Table 8 – The average results of RMS errors produced by NCNNs over 50 runs for the chlorophyll-a prediction in Lake Kasumigaura**

| Year      | Mean   | Standard deviation | Min    | Max    |
|-----------|--------|--------------------|--------|--------|
| 1983–1984 | 0.0140 | 0.0003             | 0.0135 | 0.0147 |
| 1986      | 0.0752 | 0.0061             | 0.0652 | 0.0956 |
| 1987–1992 | 0.0266 | 0.0006             | 0.0257 | 0.0283 |
| 1993      | 0.1130 | 0.0046             | 0.1052 | 0.1325 |

proposal, refers to the system's ability to deal with noises and in functioning well in the presence of minor system faults.

Although there are many papers discussing diversity and introducing diversity measures (Brown et al., 2005), there is no systematic study of different diversity measures. Little has been done to compare different diversity measures at different levels in different systems in a unified framework.

## 5.2. Coevolution and self-adaptation

In ecological communities, the environment of one species contains other species; as one species evolves through natural selection driven by its environment, it is changing the environment of others. Biological coevolution has inspired two major types of artificial coevolution: competitive coevolution and cooperative coevolution. Both of them have been used to solve complex problems (Darwen and Yao, 1997; Liu et al., 2001), but only to a limited degree of success. It is still unclear now why artificial coevolution has not worked as well as expected and certainly not as well as its biological counter-

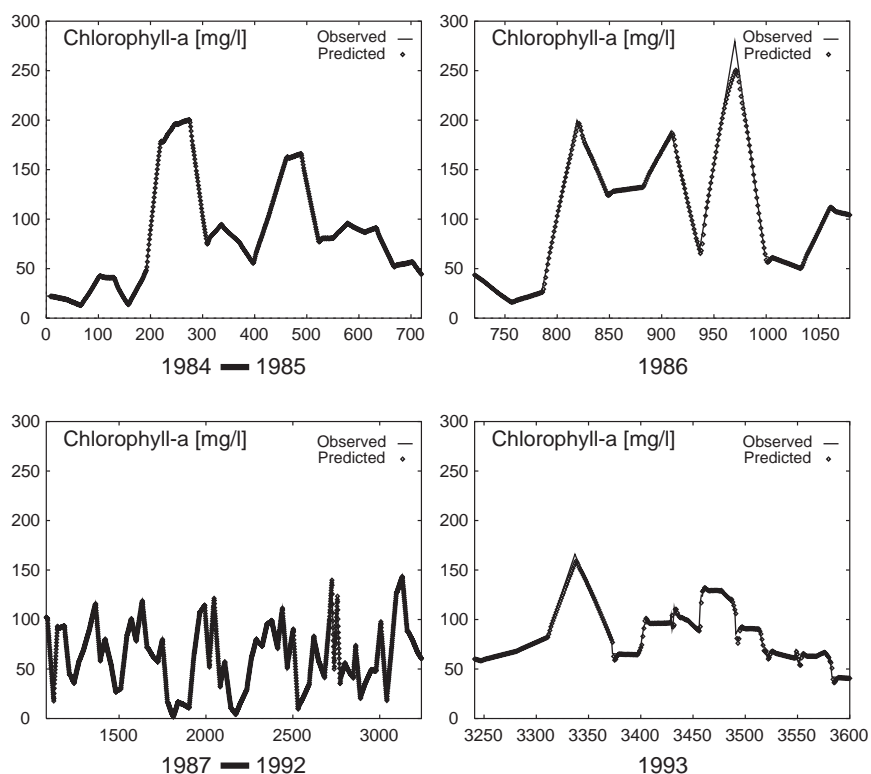
part. There are several important issues that we could explore in this area, e.g.,

1. Given a problem-solving system with many coevolving sub-systems, how does the diversity among the sub-systems influence the overall problem-solving performance?
2. What new algorithms, methods and techniques can we develop so that appropriate sub-systems can emerge more effectively from an initial random set after evolution?

## 5.3. Multi-scale and multi-level computation

An ecosystem interacts with its environment in different temporal and spatial scales: from plant growth (spatial) in a few months (temporal) to global atmosphere change (spatial) in decades (temporal). It also has very different levels within itself, from individuals in a small and local area to populations or communities in a much broader and global space. Every individual in an ecosystem has its own role and function.

Ideas and principles in biological ecosystems can inspire many different ways of designing problem-solving systems which are capable of learning and evolving appropriate levels of computation in different scales (granularities). It is interesting to study hierarchical systems that can learn and evolve its own levels and hierarchy automatically and dynamically in a changing environment. The key to deal with the complexity in complex systems is to avoid details when we do not need them. This points to a problem-solving system that can adjust its granularity of computation automatically. It can compute very quickly to find a sufficiently good,



**Fig. 5 – The observed outputs and the best NCNNs' outputs for the chlorophyll-a prediction in Lake Kasumigaura. The time span is  $\Delta t=1$ .**

but not optimal, solution. The hard question to answer is when and how the system knows to adjust its granularity of computation. There are many fundamental research questions to be addressed here, e.g.,

1. What is an appropriate architecture for such a multi-scale and multi-level problem-solving system to facilitate the learning and evolution of the system?
2. How can we represent different levels? How do different levels emerge?
3. How does a low level computation influence higher level dynamics?
4. How does the higher level dynamics guide the lower level computation?
5. How do temporal and spatial scales interact with each other? Is it possible to trade one for the other?
6. Given a practical problem how do we learn and evolve appropriate space and time scales at different levels for a problem-solving system?

## 6. Conclusion

Ecoinformatics is an extremely exciting research field. It encompasses two major types of research: one is the application of latest computational tools in solving complex problems in ecology and the other is the development of novel computational models and algorithms inspired by biological ecosystems. This paper first reviewed some of the latest computational techniques that can be applied to ecoinformatics effectively and efficiently, including data-driven evolutionary modelling, coevolutionary algorithms for large-scale optimisation and automatic problem decomposition through neural network ensembles. Then a number of new ideas for future research are presented in the area of developing novel computational models and algorithms inspired by biological ecosystems.

## REFERENCES

- Brown, G., Wyatt, J.L., Harris, R., Yao, X., 2005. Diversity creation methods: a survey and categorisation. *Information Fusion* 6, 5–20 (January).
- Chandra, A., Yao, X., 2004. DIVACE: diverse and accurate ensemble learning algorithm. *Proc. of the Fifth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'04)* Lecture Notes in Computer Science, vol. 3177, pp. 619–625. August.
- Chandra, A., Yao, X., 2005. Evolutionary framework for the construction of diverse hybrid ensembles. *Proc. of the 13th European Symposium on Artificial Neural Networks (ESANN'2005)*, pp. 253–258. April.
- Chandra, A., Yao, X., accepted for publication. Ensemble learning using multi-objective evolutionary algorithms. *Journal of Mathematical Modelling and Algorithms*.
- Darwen, P.J., Yao, X., 1997. Speciation as automatic categorical modularization. *IEEE Transactions on Evolutionary Computation* 1 (2), 101–108.
- Feller, W., 1971. 2nd ed. *An Introduction to Probability Theory and Its Applications*, vol. 2. John Wiley & Sons, Inc.
- Haykin, S., 1994. *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing Company, New York, NY 10022.
- Khare, V.R., Yao, X., Sendhoff, B., 2004. Credit assignment among neurons in co-evolving populations. *Proc. of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*. Lecture Notes in Computer Science, vol. 3242. Springer-Verlag, Berlin, pp. 882–891.
- Khosroshahi, H.G., Wadadekar, Y., Kembhavi, A., 2000. Correlations among global photometric properties of disk galaxies. *The Astrophysical Journal* 533, 162–171.
- Koza, J., Andre, D., 1999. Automatic discovery of protein motifs using genetic programming. In: Yao, X. (Ed.), *Evolutionary Computation: Theory and Applications*. World Scientific Publ. Co., Singapore.
- Lee, C.Y., Yao, X., 2004. Evolutionary programming using the mutations based on the Lévy probability distribution. *IEEE Transactions on Evolutionary Computation* 8, 1–13 (January).
- Li, B., Lin, J., Yao, X., 2002. A novel evolutionary algorithm for determining unified creep damage constitutive equations. *International Journal of Mechanical Sciences* 44 (5), 987–1002.
- Li, J., Yao, X., Frayn, C., Khosroshahi, H.G., Raychaudhury, S., 2004. An evolutionary approach to modeling radial brightness distributions in elliptical galaxies. *Proc. of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*. Lecture Notes in Computer Science, vol. 3242. Springer-Verlag, Berlin, pp. 591–601.
- Liu, Y., Yao, X., 1999a. Time series prediction by using negatively correlated neural networks. In: McKay, B., Yao, X., Newton, C., Kim, J.-H., Furuhashi, T. (Eds.), *Simulated Evolution and Learning: The Second Asia-Pacific Conference on Simulated Evolution and Learning, SEAL'98, Selected Papers*. Lecture Notes in Artificial Intelligence, vol. 1585. Springer-Verlag, Berlin, pp. 333–340.
- Liu, Y., Yao, X., 1999b. Simultaneous training of negatively correlated neural networks in an ensemble. *IEEE Transactions on Systems, Man and Cybernetics. Part B. Cybernetics* 29, 716–725 (December).
- Liu, Y., Yao, X., 1999c. Ensemble learning via negative correlation. *Neural Networks* 12, 1399–1404 (December).
- Liu, Y., Yao, X., Higuchi, T., 2000. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation* 4, 380–387 (November).
- Liu, Y., Yao, X., Zhao, Q., Higuchi, T., 2001. Scaling up fast evolutionary programming with cooperative co-evolution. *Proceedings of the 2001 Congress on Evolutionary Computation*. IEEE Press, Piscataway, NJ, USA, pp. 1101–1108. May.
- Monirul Islam, Md., Yao, X., Murase, K., 2003. A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on Neural Networks* 14 (4), 820–834.
- Recknagel, F., 1997. ANNA—artificial neural network model for predicting species abundance and succession of blue-green algae. *Hydrobiologia* 349, 47–57.
- Schnier, T., Yao, X., 2003. Using negative correlation to evolve fault-tolerant circuits. *Proceedings of the 5th International Conference on Evolvable Systems (ICES-2003)*. Lecture Notes in Computer Science, vol. 2606. Springer, Germany, pp. 35–46. March.
- Schnier, T., Yao, X., Liu, P., 2004. Digital filter design using multiple pareto fronts. *Soft Computing* 8, 332–343 (April).
- Schwefel, H.-P., 1995. *Evolution and Optimum Seeking*. John Wiley & Sons, New York.
- Törn, A., Žilinskas, A., 1989. *Global optimisation*. Lecture Notes in Computer Science, vol. 350. Springer-Verlag, Berlin.
- Yao, X., Liu, Y., 1998a. Scaling up evolutionary programming algorithms. In: Porto, V.W., Saravanan, N., Waagen, D., Eiben, A.E. (Eds.), *Evolutionary Programming VII: Proc. of the*

- 7th Annual Conference on Evolutionary Programming. Lecture Notes in Computer Science, vol. 1447. Springer-Verlag, Berlin, pp. 103–112.
- Yao, X., Liu, Y., 1998b. Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man and Cybernetics: Part B. Cybernetics* 28 (3), 417–425.
- Yao, X., Liu, Y., Lin, G., 1999. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation* 3, 82–102 (July).