

Evolving hybrid ensembles of learning machines for better generalisation

Arjun Chandra*, Xin Yao

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

Abstract

Ensembles of learning machines have been formally and empirically shown to outperform (generalise better than) single predictors in many cases. Evidence suggests that ensembles generalise better when they constitute members which form a diverse and accurate set. Additionally, there have been a multitude of theories on how one can enforce diversity within a combined predictor setup. We recently attempted to integrate these theories together into a co-evolutionary framework with a view to synthesising new evolutionary ensemble learning algorithms using the fact that multi-objective evolutionary optimisation is a formidable ensemble construction technique. This paper explicates on the intricacies of the proposed framework in addition to presenting detailed empirical results and comparisons with a wide range of algorithms in the machine learning literature. The framework treats diversity and accuracy as evolutionary pressures which are exerted at multiple levels of abstraction and is shown to be effective.

© 2006 Elsevier B.V. All rights reserved.

Keywords: Ensemble learning; Evolutionary computation; Hybrid ensembles; Multi-objective optimisation; Neuroevolution

1. Introduction

One of the main issues in machine learning research is that of generalisation. Generalisation refers to the prediction ability of a base learner (or learning machine). The better a predictor performs on unseen data, the better it is said to possess the ability to generalise. The ‘bias–variance dilemma’ is a theoretical result which illustrates the importance of generalisation in machine learning research.

An ensemble or a committee of learning machines has been shown to outperform (generalise better than) single learners both theoretically and empirically in many cases [12]. Tumer and Ghosh [46] present the formal proof of this. According to Dietterich [21], ensembles form one of the main research directions as far as machine learning research is concerned. Brown et al. [12] give an extensive survey of different ensemble methods. A theoretical

account of why ensembles perform better than single learners is also presented in [12].

Although ensembles perform better than their members in many cases, constructing them is not an easy task. As Dietterich [21] points out, the key to successful ensemble methods is to construct base models (individual predictors) which perform better than random guessing individually and which are at least somewhat uncorrelated as far as making errors on the training set is concerned. The statement essentially tells that in order for an ensemble to work properly it should have a diverse and accurate set of predictors (also mentioned in one of the seminal works on diversity in classifier ensembles by Hansen and Salamon [28]). Krogh and Vedelsby [32] formally show that an ideal ensemble is one that consists of highly correct (accurate) predictors which at the same time disagree as much as possible (i.e. substantial diversity amongst members is exhibited). This has also been tested and empirically verified [38,39]. Thus, diversity and accuracy are two key issues that should be taken care of when constructing ensembles. There exists a trade-off between these two terms as mentioned in [48].

Given that ensembles generalise better as compared to a single predictor, ensemble research has become an active

*Corresponding author.

E-mail addresses: a.chandra@cs.bham.ac.uk (A. Chandra), x.yao@cs.bham.ac.uk (X. Yao).

URLS: <http://www.cs.bham.ac.uk/~axc/>, <http://www.cs.bham.ac.uk/~xin/>.

research area and has seen an influx of researchers coming up with myriad of algorithms trying to improve the prediction ability of such aggregate systems in recent years. Brown et al. [12] give a taxonomy of methods for creating diversity. Yates and Partridge [53] show that there exists a hierarchical relationship between the ways in which diversity has and can be enforced while creating ensembles, with each method having its own diversity generating potential. Additionally, Dietterich [21] states that one of the promising and open research areas is that of combining ensemble learning methods to give rise to new ensemble learning algorithms. Moreover, incorporating evolution in segments of machine learning has been a widely studied area (e.g. evolving neural networks [8,45,52], evolving neural network ensembles [35]) with evolution treated as another fundamental form of adaptation in addition to learning [52]. Evolution makes neural systems adapt to a dynamic environment effectively and efficiently [52].

A more recent approach to ensemble learning views learning as a multi-objective optimisation problem [3]. We have proposed an algorithm called DIVACE (DIVERse and Accurate Ensemble Learning Algorithm) [16,18] which uses good ideas from Negative Correlation Learning (NCL) [34] and the Memetic Pareto Artificial Neural Network (MPANN) [1,3] algorithm, and formulates the ensemble learning problem as a multi-objective problem explicitly within an evolutionary setup, aiming at finding a good trade-off between diversity and accuracy. One very strong motivation for the use of evolutionary multi-criterion optimisation in the creation of an ensemble in both DIVACE [16,18] and MPANN [3] is that due to the presence of multiple conflicting objectives, the evolutionary approach engenders a *set* of near optimal solutions. The presence of more than one optimal solution indicates that if one uses multi-objectivity while creating ensembles, one can actually generate an ensemble automatically where the member networks would inadvertently be near optimal [13,16].

We recently attempted to integrate the aforementioned ideas into a co-evolutionary framework [17] with a view to synthesising new evolutionary ensemble learning algorithms stressing on the fact that multi-objective evolutionary optimisation is a useful ensemble construction technique. DIVACE [16,18], as an idea, gives us a perfect base on which to develop a framework wherein various ensemble methods can be combined and diversity enforcement can be tackled at multiple levels of abstraction.

The evolutionary framework gives a simple yet effective means for the development of new ensemble learning algorithms. Extending [17], we describe this framework in greater detail in sections to follow, covering ideas that lead to it and the problems it tries to account for. We also show how this framework can be instantiated and present two algorithms (where one is from [17]) resulting from it which essentially differ in their replacement schemes. These algorithms are empirically shown to be very promising. The new results presented in this paper further demonstrate the effectiveness of the framework.

2. Ensembles tackling the bias–variance dilemma and the trade-off between diversity and accuracy

In order for a learned machine/predictor to exhibit good input–output mappings for unseen data, the predictor should ‘know’ the exact mapping function it is trying to model. In practice, however, this is not possible, i.e. a predictor cannot learn the exact function it wants to learn mainly due to the dearth in the amount of data present for it to train on and due to noise in this data. The idea behind training is simply to inculcate a statistical model (a statistical oracle) within a base model manifested in the settings of its parameters and its complexity.

This statistical model can be called the network function. *The goal is to make the network function as fitting as possible to the training data and at the same time let it remain smooth enough such that if the predictor is trained using another related dataset and fitted on it, the network function does not become significantly different.* This statement exposes what is expressed by two much touted terms in machine learning research: *bias* and *variance*. Bias and variance are conflicting terms as far as achieving a good fit to the training data is concerned and there exists a trade-off between them [7], in that, if we try to reduce the bias, the variance increases and vice versa. Geman et al. [26] explain this trade-off very well and show that the expected error function can be decomposed into *bias* and *variance* components. Let the actual network function be $f(x)$ and the desired function be $\Phi(x)$ which actually is the conditional average $\langle \phi|x \rangle$, ϕ being the mean of the distribution of the output for a given x . Consider the mean square error function,

$$(f(x) - \langle \phi|x \rangle)^2 \quad \text{or} \quad (f(x) - \Phi(x))^2, \quad (1)$$

the value of which depends on the dataset used for training. The expectation operator $E\{\cdot\}$ is used to get rid of this dependence. The use of this operator gives us an error function (Eq. (2)) which is independent of the initial conditions for a predictor as well as the choice of the training set:

$$E\{(f(x) - \Phi(x))^2\}. \quad (2)$$

According to the bias–variance decomposition [26],

$$\begin{aligned} E\{(f(x) - \Phi(x))^2\} &= E\{(f(x) - E\{f(x)\})^2\} \\ &\quad + (E\{f(x)\} - \Phi(x))^2 \\ &= \text{Variance} + \text{Bias}^2. \end{aligned} \quad (3)$$

In order to help fully understand the way in which individual predictors affect the ensemble as a whole, a more general decomposition of the quadratic error function, given in [12,47], is achieved by decomposing the quadratic error into *bias*, *variance* and *covariance* terms. Since we are concerned with the error expectation on future/unseen data points and need to be consistent with the bias–variance decomposition literature, we need to view the expected value of the error over all possible choices of

datasets on which the predictors could be trained on and over all possible parameter initialisations.

In case of an ensemble, let $f(x)$ be the ensemble output. Considering simple averaging as our combination rule,

$$f(x) = \frac{1}{M} \sum_{i=1}^M f_i(x),$$

where M is the number of networks in the ensemble. For simplicity, we will write

$$f(x) = \frac{1}{M} \sum_i f_i. \quad (4)$$

The variance component of Eq. (3) can be further subdivided into two parts: variance and covariance. Substituting Eq. (4) into the variance component of Eq. (3) we have

$$\begin{aligned} & \text{Variance(ensemble)} \\ &= E \left\{ \left(\frac{1}{M} \sum_i f_i - E \left\{ \frac{1}{M} \sum_i f_i \right\} \right)^2 \right\} \\ &= \frac{1}{M^2} E \left\{ \sum_i \sum_{j=1, j \neq i}^M (f_i - E\{f_i\})(f_j - E\{f_j\}) \right\} \\ &\quad + \frac{1}{M^2} E \left\{ \sum_i (f_i - E\{f_i\})^2 \right\} \\ &= \frac{1}{M^2} \text{Covariance} + \frac{1}{M^2} \text{Variance}. \quad (5) \end{aligned}$$

We also use a variant of the *Covariance* term in DIVACE [16,18]. This variant was initially used in one of the established ensemble learning algorithms called Negative Correlation Learning (NCL) proposed by Liu and Yao [34] as a regularisation term.

The bias–variance–covariance decomposition can now be expressed as

$$\begin{aligned} & E \left\{ \left(\frac{1}{M} \sum_i f_i - \Phi(x) \right)^2 \right\} \\ &= (E\{f(x)\} - \Phi(x))^2 \\ &\quad + \frac{1}{M^2} E \left\{ \sum_i \sum_{j=1, j \neq i}^M (f_i - E\{f_i\})(f_j - E\{f_j\}) \right\} \\ &\quad + \frac{1}{M^2} E \left\{ \sum_i (f_i - E\{f_i\})^2 \right\}. \quad (6) \end{aligned}$$

Eq. (6) expresses the fact that the quadratic error of the ensemble depends on bias, variance and also on the relationships between individual members of the ensemble. The *covariance* term here can be said to indicate the *diversity* or *disparity* between the member networks as far as their error estimates are concerned.

Hence, it is believed that the more diverse the individual members an ensemble has, the less correlated they would be, which seems obvious. This suggests that the covariance term should be as low as possible. The lower the covariance term, the less the error correlation amongst the networks, which implies reduced error and better performance at the ensemble level. This is the main reason why *diversity* in neural network ensembles is extremely important. This is also true for any kind of ensemble, be it one having support vector machines as its members, decision trees, or for that matter, a mixture of various types of learning machines. A thorough discussion of diversity with regard to neural network ensembles is covered in [11,12].

Apart from this, Krogh and Vedelsby [32] show that the mean square error/quadratic error of the ensemble is guaranteed to be less than or equal to the weighted average quadratic error of the member networks, proving the existence of a trade-off between accuracy and diversity.

Mathematically,

$$\begin{aligned} \sum_i w_i (f_i - \Phi)^2 &= \sum_i w_i (f_i - f_{\text{ens}})^2 + (f_{\text{ens}} - \Phi)^2 \\ &\Rightarrow (f_{\text{ens}} - \Phi)^2 = \sum_i w_i (f_i - \Phi)^2 \\ &\quad - \sum_i w_i (f_i - f_{\text{ens}})^2. \quad (7) \end{aligned}$$

Here, f_{ens} is the ensemble output given by

$$f_{\text{ens}} = \sum_i w_i f_i,$$

where f_i is the output of network i ; f_{ens} is a convex combination, i.e. $\sum_i w_i = 1$ and $\forall i, w_i \geq 0$; Φ is the desired output.

In Eq. (7), the second term on the right hand side (i.e. $\sum_i w_i (f_i - f_{\text{ens}})^2$) will always be greater than or equal to zero. This is often called the ambiguity term and it tells how different individual members within the ensemble are for some test pattern (data point), i.e. emphasises diversity. It means that if this term is somehow made large, the left hand side will become small, i.e. the error estimate of the ensemble for a given data point will decrease. In other words, if we have more diversity in the ensemble, the accuracy of the combined predictor will increase. It can also be seen that if we try to maximise this term i.e. cross some limit (let us call it a trade-off line), the first term (i.e. $\sum_i w_i (f_i - \Phi)^2$) may also increase, which then would cancel out the whole effect caused due to the amplification of the ambiguity term. Here, the first term signifies the accuracy of an individual for some test pattern. *The second term signifies that we should have diverse members in the ensemble but there is a limit/trade-off line crossing which would make the individual predictors less accurate.* This is what is often referred to as the *trade-off between diversity and accuracy* in ensembles, a problem which we tried to tackle in DIVACE [16,18] and which we try to tackle here as well in a similar fashion.

2.1. Ensembles reducing bias

As is evident from the previous discussion, ensembles reduce the expected error (Eq. (2)) by reducing the variance term of Eq. (3). However, Valentini et al. [48] refer to the fact that ensembles reduce either one or both terms of Eq. (3), i.e. they may reduce bias too. Some ensemble methods have been shown to have a greater effect on the reduction in the expected error. This is due to them acting on both the bias and variance terms of Eq. (3). These include two of the most widely studied ensemble approaches, namely, bagging [9] and boosting [24,36].

In light of this evidence, we thought it would be worth experimenting with an ensemble method which uses these two methods (i.e. bagging and boosting) in some way to exploit the error reducing properties exhibited by both. As will be shown later, we try to put these positive aspects of bagging and boosting to good use in our approach (both our framework and the algorithm resulting from it). Since a combination of methods helps enhance performance as stated in [21], this gives us further motivation in combining bagging and boosting.

2.2. Diversity helps

From the bias–variance–covariance decomposition (Eq. (6)) we know that ensembles usually try to lower the expected error by having members which are uncorrelated in producing errors (lowering the covariance term). This makes the variance term of the expected error function (Eq. 3) low, hence reducing error or enhancing performance of the predictor formed.

Having uncorrelated failures suggests that the ensemble should have members which individually work well on different parts of the training set. Making errors on different, rather disjoint parts of the training set would result in a situation where it is possible to get the correct output for each training pattern when the outputs of individuals are combined.

Sharkey et al. [43] suggest that for an ensemble to have good diversity, the component predictors should show different patterns of generalisation (i.e. have uncorrelated failures with respect to the test set). Uncorrelated failures with respect to the test set is achieved by estimating uncorrelated failures on the training set and this is what most ensemble methods try to do.

There is a large body of research discussing holistic classification schemes for ensemble learning methods. Valentini et al. [48] categorise ensemble methods as being either generative or non-generative depending on whether they actively or passively boost the accuracy and diversity of the base learners. The usual stance taken in some schemes is of using diversity as the categorising criterion. Sharkey [42] presents one such scheme. However, Brown et al. [12] recently proposed a new classification scheme which they feel covers a majority of the ensemble methods developed to date. This scheme includes, to our knowledge,

more ensemble methods than any other scheme and so we took the ideas expressed in it while constructing our framework as will be seen shortly (Section 4).

Seeking a predictor from within a hypothesis space and using a set of predictors thus found results in the formation of an ensemble. Brown et al. [12] use this idea as the basis for classifying ensemble methods and state that in order to construct an ensemble, predictors from different points in the hypothesis space are desirable. This is determined by the way in which this space is explored. Accordingly, diverse ensembles can be constructed by making the components learn in three main ways as given in Table 1.

Apart from the above discussed classification scheme, Yates and Partridge [53] came up with a scheme which puts diversity generating methods into various levels depending on the efficacy of each in enforcing diversity within an ensemble. According to them, methodological diversity is the most effective diversity generating method. This is followed by training set structure, architecture of the learning machine and initial conditions for the base learners in this order. Fig. 1 shows this ordering.

Building on the efficacy of methodological diversity, Wang et al. [49,50] developed a method of constructing ensembles using both neural networks and decision trees as base models. Also, Langdon et al. [33] have tried combining neural networks and decision trees. Woods et al. [51] go one step further by constructing an ensemble using four methodologically different types of learners. These heterogeneous ensembles are commonly known as hybrid ensembles.

As can be seen in Fig. 1, this hierarchy considers two out of the three main diverse ensemble constructing methods from the scheme shown in Table 1. In our framework, we want to consider enforcement of diversity at all these levels, incorporating class C from Table 1 as well, thus taking care of both the classification scheme and the diversity hierarchy.

As mentioned in [12], there is a disappointingly small number of investigations into the field of hybrid ensembles. This paper proposes a new evolutionary approach to hybrid ensembles.

Table 1
Ensemble classification scheme from Brown et al. [12]

Class	Description
A	by making the component learners start the learning process with different initial conditions. They call this the “Starting point in hypothesis space” [12] class.
B	by manipulating the training set used, using different architecture or using different learning algorithms to train the components. This is called “Set of accessible hypotheses” [12].
C	by modifying the trajectory used by the components in the hypothesis space using penalty function (as a regularisation term) methods and global search methods such as evolutionary algorithms. More formally referred to as “Traversal of hypothesis space” class as per [12].

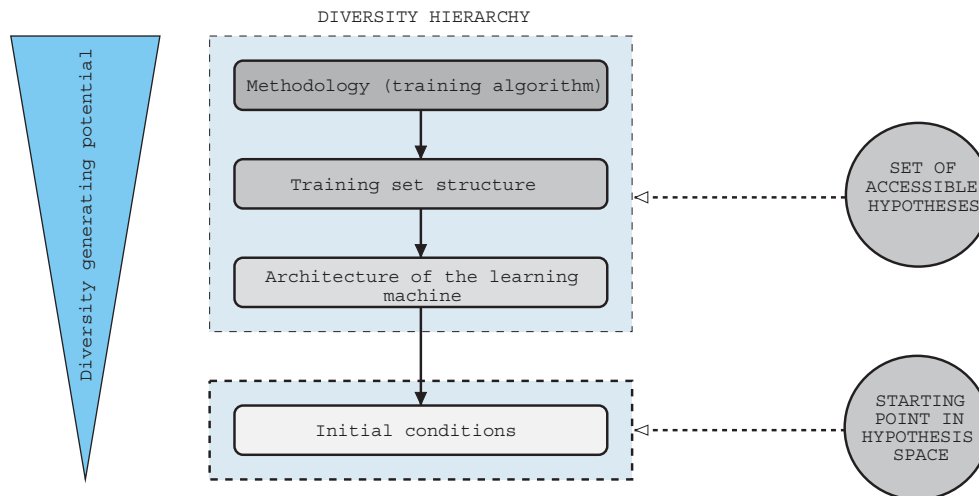


Fig. 1. Diversity hierarchy [14,53].

3. Multi-objective evolution for ensembles

Multi-objectivity in ensembles, as an area of research, has not been explored extensively yet. According to our knowledge, the idea of designing neural networks within a multi-objective setup was first considered by Kottathra and Attikiouzel [31] where they used a branch and bound method to determine the number of hidden neurons (the second objective being the mean square error) in feed forward neural networks. Recently, Abbass [4] proposed an evolutionary multi-objective neural network learning approach where the multi-objective problem formulation essentially involved setting up of two objectives viz. complexity of the network and the training error (quadratic error/mean square error). The network complexity here could mean the number of synaptic weights, number of hidden units or a combination of both. An algorithm called MPANN was proposed which uses Pareto differential evolution [5]. MPANN was later on considered [2,3] for learning and formation of neural network ensembles, albeit, with a different multi-objective formulation (as opposed to that in [4]). Based on multi-objective evolutionary optimisation, taking inspiration from MPANN [2,3], and exploiting the main idea of NCL [34] (which belongs to class C of the scheme discussed in the previous section) we recently proposed DIVACE [16,18].

The main reason for using a multi-objective evolutionary approach to designing ensembles is that multi-objectivity enforces the search/optimisation process (the search process here being finding neural networks with good overall performance) to yield a set of near optimal solutions instead of just a single solution. Getting a set of solutions essentially means that we get a set of near optimal neural networks. These near optimal neural networks could in turn be used as members of an ensemble. With a population based approach we will, in essence, be generating a set of networks and the underlying multi-objective framework would take care of the selection of a

set of near optimal solutions/networks from the population. The whole process of generating a pareto set of neural networks which could be used as members of an ensemble will be automatic as the whole population would, with the passage of time, move towards the pareto front. Thus, the idea of using multi-objective evolutionary algorithms for the purpose of designing neural network ensembles is very promising.

The main problem in actually using such an approach is the formulation of the multi-objective optimisation problem such that not only the final ensemble created have accurate members but the members also be uniformly distributed on the pareto optimal front, i.e. diversity is catered for. Remarkably, for multi-objective optimisation to be effective, the optimisation process should lead to convergence to the pareto optimal front while at the same time maintaining as diverse a distribution of solutions as possible on the pareto front [20]. A striking parallel between multi-criterion optimisation and the necessity of having diverse enough members within an ensemble is evident. Hence, formulating a problem properly would surely do justice to both (multi-criterion optimisation and ensembles as disparate yet related computational paradigms).

Thus, incorporating the idea of multi-objective evolutionary optimisation into our framework for designing ensembles is an attractive proposition. The next section takes a closer look at the framework and enucleates the intricacies therein.

4. A framework for the evolution of hybrid ensembles

Designing hybrid ensembles as a field of research is still in its infancy. There are many issues which one should consider while tackling this relatively new paradigm in ensemble research.

One reason for the lack of substantial literature in this area or the relative inactivity could be attributed to the fact

that hybrid ensembles as the name implies, deal with the combination of base learners that are trained using different training methodologies (algorithms). Brute force or even a search through a predefined search space (as some researchers have tried [27,49–51]) could result in a hybrid ensemble but would have a big disadvantage.

The hypothesis space for even a single predictor is very large and machine learning algorithms usually end up in some local optima while searching for a good hypothesis. Searching for an optimum for an ensemble is even more challenging.

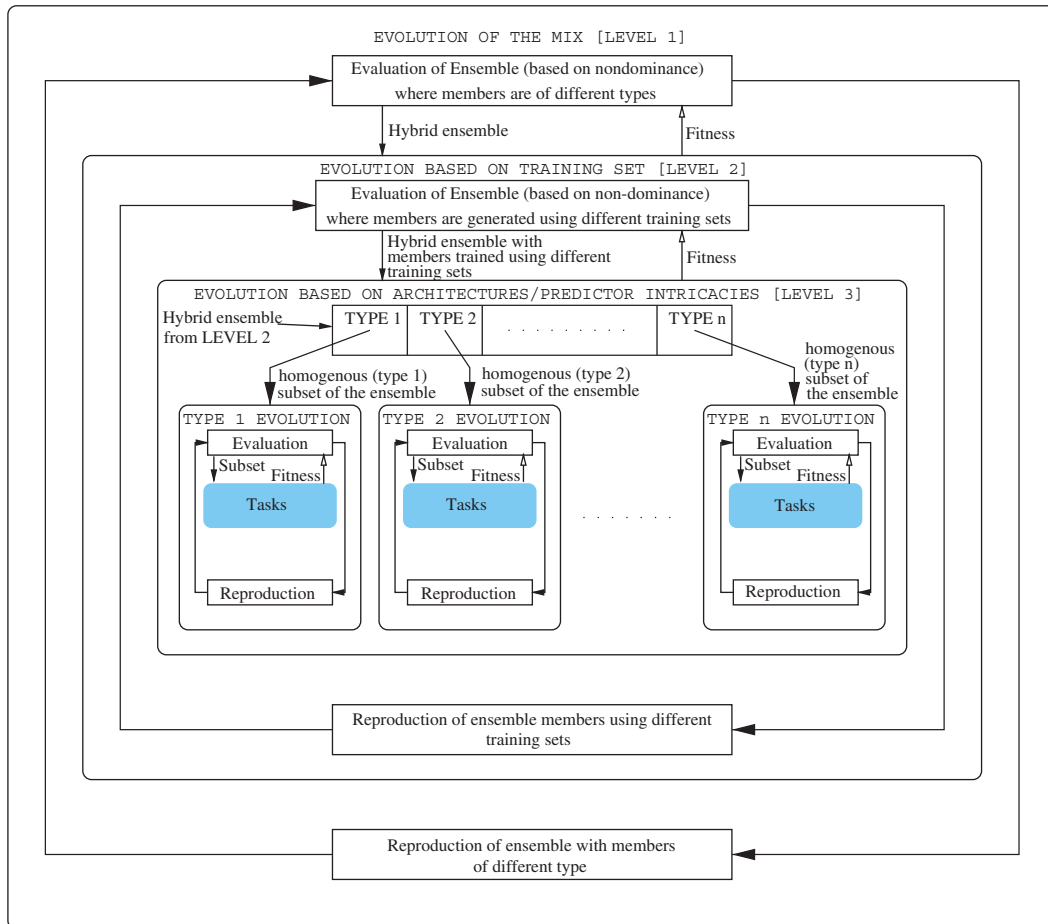
Some researchers have suggested using evolutionary algorithms [33] to evolve the mixture of individual predictors (mix), which intuitively looks very plausible but the types of individuals and learning algorithms have to be determined beforehand.

A good mix of learners can only be good with respect to the decision combination strategy utilised. Decision combination is another active area of research when it comes to multiple classifier systems or ensembles in general. Empirical studies on previously proposed combination strategies are presented in [29].

A possible evolutionary framework for the construction of diverse hybrid ensembles can be described by Fig. 2 [14,17]. As can be seen, there are three levels of evolution present. First is the evolution of the mix, i.e. evolving the mixture of the various types of predictors. Second, we consider the evolution of ensembles based on the structure of the training set (given the mix). A process similar to the original DIVACE [16,18] forms the third and final evolutionary level.

The framework shows that the hybrid ensemble at Level 2 will have subsets of different types of predictors. These subsets can, in themselves, be considered as homogeneous ensembles and evolved in accordance with DIVACE [16,18], keeping the other subsets fixed. Level 3 can be called as the DIVACE [16,18] stage where reproduction depends on the evolutionary factor(s) chosen for a given predictor type. The factors could be architectures or weights in case of NNs, kernel functions in case of SVMs, or architectures of RBFNs, etc.

Level 3, due to its subset evolutionary process, enforces competition between the various subsets. This competition makes the framework as a whole model a co-evolutionary



NOTE: Fitness consists of 2 values for each member in the ensemble/subset viz. Diversity and Accuracy

Fig. 2. The proposed framework [14,17].

Table 2
Problems to solve and a solution for each within the proposed framework

Problem	Solution
From diversity hierarchy	
Methodological difference	Level 1 evolution
Training set structure	Level 2 evolution
Predictor intricacies	Level 3 evolution
Initial conditions	Implicitly catered for
Ensemble method classification scheme of Brown et al. [12]	
Starting point in hypothesis space	Implicitly catered for
Set of accessible hypothesis	All three levels
Traversal of hypothesis space	The framework in itself is evolutionary and evolution is one way by which one can traverse the hypothesis space. Penalty approaches like NCL also help traverse the search space and we have borrowed the idea of negative correlation in the formulation of our multi-objective problem as well.
Issues with heterogeneous ensembles	
Types	Predetermined
Size	Evolutionary process can have a size control mechanism or size can be fixed beforehand.
Mixing strategy	Level 1
Decision combination	Based on types of predictors used and training set used to train each member—we use majority voting (in the resulting algorithm).
Predictor intricacies	Level 3
Other important issues	
Diversity-accuracy trade-off	Function evaluation (all levels)
Combination of ensemble methods	Population initialisation and Level 2 offspring generation (in the resulting algorithm)

learning strategy where subspecies (subsets) compete with each other to stay in the ensemble. Additionally, these very species cooperate at the immediate higher level (Level 2) and compete again at Level 1. *The framework therefore embodies both competitive and cooperative co-evolution within a multi-objective and multi-level evolutionary setup.* The choice on the placement of these levels essentially depends on the prior knowledge available. However, the above mentioned co-evolutionary theme would be most effective if we keep this very ordering in the levels or at least let the innermost level stay where it is as it makes more sense to have a hybrid set of predictors and then let the subsets compete with each other than to enforce competition without having a mix.

Essentially, there is a range of problems the framework tries to solve and these are enumerated in Table 2 together with the ways these have been addressed.

The framework incorporates a majority of the ideas expressed by researchers in order to construct ensembles which have both diverse and accurate members so as to have good generalisation capabilities. We can say that the framework represents a generic model from which new hybrid ensemble construction algorithms can be developed. It represents a class of ensemble learning methodologies aimed at enforcing diversity and accuracy explicitly within an evolutionary framework. In the next section we will instantiate the framework giving a possible algorithm.

5. Instantiation of the framework

Here we present our algorithm, i.e. DIVACE-II (as a successor of DIVACE [16,18]), which can be thought of as being one instance of the framework. In DIVACE-II, we try to incorporate all the levels mentioned in the framework presented in the previous section. However, one should limit the ensemble construction approach to as fewer levels as possible depending on domain knowledge due to the computationally intensive nature of evolutionary methods. We model all three levels in our algorithm mainly to test the effectiveness of the framework.

We use a technique similar to AdaBoost [6] while initialising the predictor population as well as generating offspring. The idea is to generate a new training set, $\frac{1}{4}$ th of which is generated in a manner similar to AdaBoost and the rest of the instances are chosen randomly from the original training set. Lets call this as the *BagBoost* procedure. Following is the DIVACE-II algorithm:

DIVACE-II Algorithm (Explorative replacement)

Step I: Initialise the population of predictors¹ using Bagging [9] and *BagBoost*.

Step II: Perform k -means clustering [22] using the Euclidean distance (with respect to the failure patterns² of the predictors on the original training set) to form M^3 clusters and select the best⁴ predictor from each cluster to form the initial ensemble.

Step III: Repeat until termination conditions (a certain number of generations in our case) are met.

(1) Preserve the elite: archive the current ensemble if it dominates the previous best ensemble (based on training and test accuracies).

¹Population contains p number of NNs, p number of SVMs and p number of RBFNs, where $p = 20$. 10 individuals generated using Bagging and 10 using *BagBoost* for each predictor type.

²A failure pattern is a string of 0s and 1s indicating success or failure of the learning machine on the training instances in the original training set.

³ $M = 25$.

⁴Best/worst individual/predictor wherever mentioned is in terms of accuracy on the original training set.

- (2) Evaluate the individuals in accordance with the two objective functions (accuracy and diversity)⁵ and label the non-dominated set as in [16].
- (3) All non-dominated individuals are selected as parents. Training examples mis-classified by the parents have their associated probability values increased sequentially as in AdaBoost.
- (4) If non-dominated individuals = total number of individuals in the current ensemble then goto (5).
 - Generate an offspring pool⁶ using the training set generated in the previous step and applying *BagBoost* for all the types of predictors being used.
 - Cluster the offspring pool using *k*-means clustering where the number of clusters is decided by the number of dominated individuals.
 - Replace the individuals that are dominated with the best individual from each cluster while making sure that only the individuals which are common with respect to types are replaced. This replacement strategy ensures equal representation of each type of predictor.
 - Goto (6).
- (5) Generate an offspring pool⁷ using the training set generated in (3) and applying *BagBoost*, the size of which is equal to the number of types of predictors being used. Replace the worst individual (in the population and one which is not in the ensemble) of the same type (as the best individual in this pool) with the best from this pool.
- (6) Re-cluster the population with the number of clusters equal to the size of the ensemble. This is done to ensure that there is only one member from each cluster present in the ensemble at all times. The best member is selected to be included in the new ensemble.
- (7) (*Optional*) Level 3 evolution. Perform DIVACE for the various subsets in the new ensemble which subsequently gives rise to a newer ensemble.

Step IV: Use the archived ensemble as the final hybrid ensemble.

There can be two ways in which replacement of dominated individuals from the ensemble and hence the population, with the newly generated offspring, can take place. One is an explorative method which essentially was presented in the outline of the algorithm above. The other replacement strategy is a restrictive approach in that it does not involve re-clustering after replacement. To give a brief insight into the modified replacement scheme, the *restrictive algorithm* will now be outlined.

⁵Multi-objective formulation similar to that in DIVACE [16,18] where accuracy was formulated as the mean squared error and diversity as the correlation penalty function in [35].

⁶Pool contains *q* number of NNs, *q* number of SVMs and *q* number of RBFNs, where *q* = 15.

⁷Pool contains 1 NN, 1 SVM and 1 RBFN.

DIVACE-II Algorithm (Restrictive replacement)

Step I: Initialise the population of predictors using Bagging [9] and *BagBoost*.

Step II: Perform *k*-means clustering [22] using the Euclidean distance (with respect to the failure patterns of the predictors on the original training set) to form *M* clusters and select the best predictor from each cluster to form the initial ensemble.

Step III: Repeat until termination conditions (a certain number of generations in our case) are met.

- (1) Preserve the elite: archive the current ensemble if it dominates the previous best ensemble (based on training and test accuracies).
- (2) Evaluate the individuals in accordance with the two objective functions (accuracy and diversity) and label the non-dominated set as in [16].
- (3) All non-dominated individuals are selected as parents. Training examples mis-classified by the parents have their associated probability values increased sequentially as in AdaBoost.
- (4) If non-dominated individuals = total number of individuals in the current ensemble then goto (5).
 - Generate an offspring pool using the training set generated in the previous step and applying *BagBoost* for all the types of predictors being used.
 - Cluster the offspring pool using *k*-means clustering where the number of clusters is decided by the number of dominated individuals.
 - Replace the individuals that are dominated with the best individual from each cluster.
 - Goto (6).
- (5) Generate an offspring pool using the training set generated in (3) and applying *BagBoost*, the size of which is equal to the number of types of predictors being used. Replace the worst individual (in the ensemble), without any regard to the type of the individual, with the best from this pool if this best individual dominates (in terms of both accuracy and diversity) it.
- (6) (*Optional*) Level 3 evolution. Perform DIVACE for the various subsets in the new ensemble which subsequently gives rise to a newer ensemble.

Step IV: Use the archived ensemble as the final hybrid ensemble.

5.1. Algorithm details

5.1.1. Choice of parameters

A critique on the choice of parameters viz., the types of predictors used, the size of the ensemble and the decision combination strategy, follows:

Types of predictors. There have been very few studies, as per our knowledge, that take the types of predictors into

account while considering the design of hybrid ensembles. This is done arbitrarily in most cases. Yates and Partridge [53] used neural networks and radial basis function networks together in an ensemble and showed that type differences result in greater diversity and hence better performance of the ensemble as compared to architecture differences. Decision trees and neural networks have also been combined to form hybrid ensembles in [33,49,50] but still there was no proof as such to determine the effectiveness of using just decision trees and neural networks and not any other kind of learner i.e. the choice was arbitrary.

Specifically, arbitration in this area i.e. selection of types of predictors, can hardly be avoided mainly because of the nature of the field (machine learning). There are myriads of machine learning algorithms and new algorithms are emerging as well. Therefore, choosing the types of predictors is best left as an arbitrary decision. In the algorithm that results from our framework, we basically use three types of learners viz. neural networks, support vector machines and radial basis function networks.

Size of the ensemble. Bagging [9] and a variant of boosting [24] are two approaches we have used in our algorithm and it has been considered by Opitz and Maclin in [38] that the greatest reduction in error (reduction in error on the test set i.e. increase in generalisation ability) using these methods comes with the first few predictors. Many empirical results to prove this fact were also shown. They use ensembles of decision trees and neural networks to report their findings on the appropriate size of an ensemble. They mainly found that all methods they tested (bagging and boosting with both neural networks and decision trees) showed little reduction in the test error with an ensemble size crossing 25. Since we are using bagging and a variant of boosting too in our algorithm, we consider an ensemble size of 25.

Decision combination. Although predictors of different types complement each other as far as the ensemble performance is concerned [29], combining potentially conflicting decisions is somewhat problematic [29].

As mentioned in [29], strengths and weaknesses of individual predictors should be taken into account in any combination strategy. Many combination strategies have been surveyed in [29]. Woods [51] also refers to some popular combination approaches which try to tackle this issue. As per [51,44], there are essentially two ways in which decisions by individual predictors can be combined to get an ensemble output. One is a cooperative strategy and the other, a competitive approach. Cooperative approaches generally involve fusion of the outputs from ensemble members (e.g. majority voting) whereas competitive strategies deal with a selection mechanism wherein the best individual is chosen as the one representing the ensemble output (e.g. selection by local accuracy [51], winner-takes-all etc.).

Also, one major issue when combining learners of different types is the representation of outputs/decisions

made by the members mainly because the decision combination strategy works on these very outputs. The representation should be flexible enough such that it can take care of different types of predictors and at the same time present the combination strategy with some useful information. Further, it should represent the same information regardless of the type of predictor for the decision combination strategy to work on it in an effective and efficient manner.

In our approach, we try to have a generic format for the outputs of individual predictors, thus facilitating the use of a simple combination strategy. The main strategy we use is a majority vote of the outputs of the individual predictors. This is essentially done to cater for ordinal outputs from the support vector machine predictors. Consequently, the outputs from the neural networks and radial basis function networks in the ensemble need to be snapped to ordinal outputs depending on a certain threshold value. We mainly tested our approach on binary classification problems which led us to set the threshold value to 0.5, where 0 and 1 represent the two classes.

5.1.2. Representation

Evolving a heterogeneous population entails representing it in a manner which is consistent with all types of individuals in this population. However, having the same representation for functionally different types of individuals/predictors is very difficult. We therefore use a direct representation for all types of predictors. Even though they are functionally different, there still is some common information between them. We call this information as book-keeping information which needs to be in place for each member in the population and so we chose to represent an individual the way shown in Fig. 3.

Book-keeping information is essentially used to speed up the whole evolutionary process and is basically a storage space for the information used during the various evolutionary stages of the algorithm. Predictor specific information, as the name suggests, contains information which is used to represent the individuals internally (in memory) and has the essential elements detailing the structure and function of the predictors.

5.1.3. Population initialisation

We use bagging and *BagBoost* to initialise the population of learners which are then evolved. It is a well known fact that boosting is an ensemble method concerned with generating hypotheses sequentially (step-wise) and enforcing diversity explicitly based on sampling training instances (for the new hypothesis) according to a probability distribution where this probability distribution depends on the mis-classifications of examples exhibited by the hypothesis generated in the previous step. A hypothesis essentially works half the time on mis-classified examples of the previous hypothesis [40].

However, boosting (for instance, AdaBoost [25]) has often been shown to overfit in situations where the training

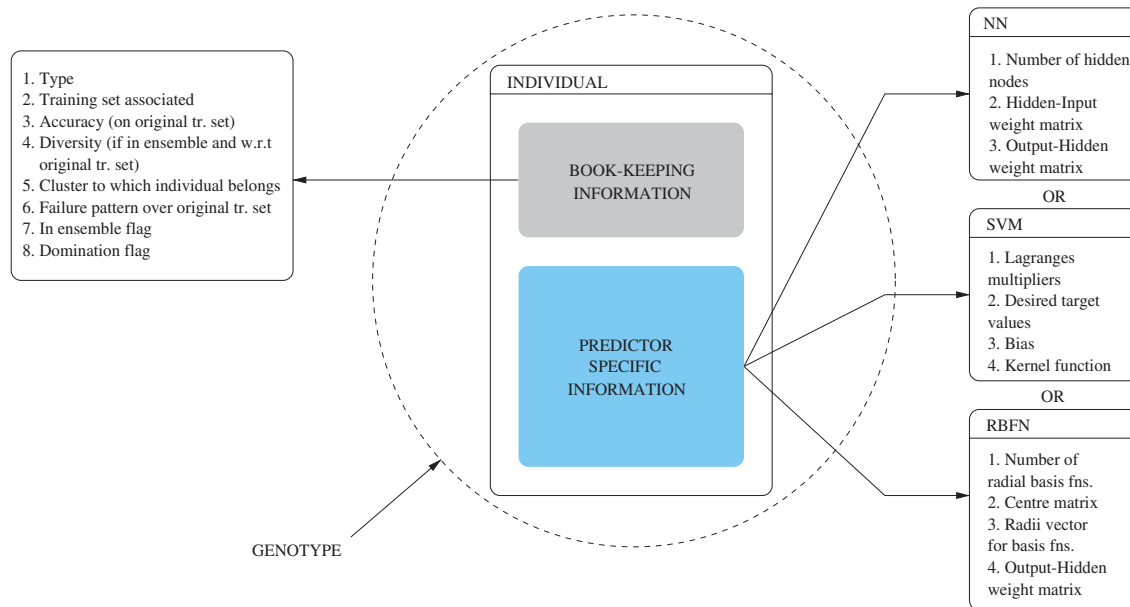


Fig. 3. Genotypic representation of an individual [14].

data is noisy [38]. Noisy data is often difficult to classify and so AdaBoost assigns high weights (sampling probability values) to these examples which leads to overfitting. We use a strategy which reduces the sampling probability through a deterministic manipulation of the sampled space, i.e. the training set resulting from the sampling of the training instances for the next hypothesis in the ensemble construction process, and hence try to avoid overfitting. We essentially sample $\frac{1}{4}$ th of the training set using probability values that are based on AdaBoost (boosting in general) and the rest is sampled randomly (as done in bagging [9]).

In other words, we reduce the possibility of overfitting by making the predictor work a fraction (given by $\frac{1}{4} \cdot \frac{1}{2} = \frac{1}{8}$) of its time on the training instances mis-classified by the previous hypothesis and the rest on the whole training set. The fraction $\frac{1}{8}$ th here is due to the fact that using simple boosting (AdaBoost), a predictor spends $\frac{1}{2}$ the time learning mis-classified samples (mis-classified by the previous hypothesis), and with *BagBoost*, working only $\frac{1}{4}$ th the time like boosting would further decrease the probability of the new hypothesis working on the mis-classified samples to $\frac{1}{8}$ th. Although arbitrarily chosen, this relieves us from using a weighted majority-voting-like strategy (which is the norm in boosting) to some extent.

We need to promote both accuracy and diversity and following the exact procedure as AdaBoost would mainly take care of diversity (AdaBoost combines weak learners but we want strength too to explore the effects of mixing ensemble learning methods). Hence, a mixture of boosting and bagging is one possibility (may not be the best one) by which we can promote both accuracy and diversity while at the same time using a simple decision combination strategy, i.e. a simple majority vote.

5.1.4. Fitness evaluation

We evaluate the individuals using a multi-objective evaluation function. The two objectives on which to optimise the performance are accuracy and diversity. Accuracy is formulated as the mean squared error and diversity as the correlation penalty function in [35]. The actual metrics can be found in [16,18]. DIVACE [16,18] was shown to achieve a good trade-off between diversity and accuracy utilising this multi-objective evaluation approach and we use it here as well.

5.1.5. Evolution of the mix (Level 1)

One could argue that to get a good mix of individuals to form an ensemble (i.e. for Level 1 evolution) we could sample individuals from the population and form ensembles (mixes), followed by evaluating each mix and selecting the best from this new ensemble population (mixes). However, evolution at all levels (from the framework) would be very expensive computationally—fitness evaluations can be expensive for instance. Hence, we follow the idea expressed by Jin and Sendhoff [30] where they stated that clustering can reduce the number of fitness evaluations by limiting the evaluations to just a chosen few solutions (ensembles in our case). They use a k -nearest-neighbour method to group solutions and select the solution closest to the cluster centres to be evaluated using the expensive fitness function. We use the k -means algorithm to cluster the population and select the best individual from each cluster to form the ensemble which goes through the evolutionary phases in the algorithm (DIVACE-II) later on. We are essentially reducing the number of fitness evaluations to just one solution (ensemble) for the Level 1 evolutionary stage in the framework.

5.1.6. Evolution at the training set level (Level 2)

The mating pool contains the individuals which are in the non-dominated set. We use a special kind of crossover operator that is based on failure patterns of the parents on the original training set. All the members in the non-dominated set are considered as the parents for all the offspring generated. Offspring generated also take up the role of parents to generate further offspring.

The main step in this crossover operator is the formation of a training set using all the non-dominated individuals in the current ensemble. This process is similar to *BagBoost* (which was described earlier). We start with a distribution of weights (each set to $1/T_size$) on the training set which are just the sampling probability values, where T_size is the size of the training set. We increase the probability values of instances mis-classified by all non-dominated individuals (and already generated off-spring) sequentially as done in AdaBoost [25] such that the sum of the probability values remains 1 [40].

The new training set is generated using the resulting probability distribution in that $\frac{1}{4}$ th of the samples are selected using this distribution and the rest selected randomly from the original training set. This new training set is then used in a manner similar to *BagBoost*. The parent pool predominantly changes or increases by one member (offspring) and the distribution modified to generate further offspring (which are further included in the parent pool) and the process continues. It should be noted here that the three offspring subpopulations generated start with the same distribution so there are three parallel offspring generation processes running, one for each type of predictor.

Another point to make is that, it may so happen that the number of non-dominated individuals is the same as the size of the ensemble. In this case only three offspring are generated (i.e. the offspring pool consists of one member of each type of predictor being considered). This operator is trying to make an offspring pool that not only has diverse and accurate members in itself but also with respect to the non-dominated set.

The offspring pool has one subpool of NNs, SVMs and RBFNs each. We need to find individuals in this pool which can replace the dominated individuals in the current ensemble to form the new ensemble. This is done by clustering the offspring pool using Euclidean distance as the distance measure with respect to the failure patterns exhibited by each individual in this pool. The number of clusters is decided by the number of dominated individuals present in the current ensemble. Once clusters are formed, the best individual in each cluster (best in terms of training accuracy) is selected and put into a candidate pool which is the pool from which individuals will be taken to replace the dominated individuals in the current ensemble. In the case with only three individuals in the offspring pool, the best (in terms of training accuracy) individual is taken into the candidate pool (which now has only one individual).

The replacement (of non-dominated individuals by individuals in the candidate pool) schemes mainly differ in the fact that explorative replacement gives equal opportunity to each type of predictor to be selected from the main population at all times. In essence, we evolve the population as a whole in the hope of finding species (a representative of each forming part of the ensemble) whereas the restrictive approach is primarily focussed on evolving just the ensemble in not having a clustering (Level 1 evolution) phase after the actual replacement and is expected to be faster.

5.2. Evolution based on predictor intricacies (Level 3)

This is an optional step in the algorithm and essentially models DIVACE [16,18] explicitly. We could consider evolving all the homogeneous subsets, i.e. for NNs, SVMs and RBFNs. For instance, in case of NNs, we could evolve the subset exactly the way as it is done in DIVACE [16,18]. For SVMs, one could consider generating offspring such that they would have a kernel function which is the same as the most common kernel in the non-dominated set. For RBFNs, we could use the same strategy as that in DIVACE [16,18]. We could even consider evolving the architectures of NNs and RBFNs by generating offspring using the non-dominated set in a way such that useful information present in the architectures is propagated. This level of evolution, as has been explained while discussing the framework, is aimed at enforcing competition between the different subsets/species (with respect to predictor type) and makes the algorithm model co-evolution. Experimenting with this step will be the future work in our research.

6. Experimental results and discussion

DIVACE-II was tested on two benchmark datasets (Australian credit card assessment dataset and Diabetes dataset), available by anonymous ftp from ice.uci.edu in /pub/machine-learning-databases. Apart from the two versions of DIVACE-II discussed above, we also experimented with the multi-objective formulation. Pairwise Failure Crediting (PFC) [18] was recently proposed as a diversity measure which credits individuals in the ensemble with differences in the failure patterns, taking each pair of individuals and accruing credits in a manner similar to implicit fitness sharing [19,23]. We consider results obtained using this and the NCL penalty function term diversity measures with both the explorative and restrictive versions of DIVACE-II. In the discussion to follow, the different versions of DIVACE-II are denoted as

- (1) DIVACE-II: Explorative version with NCL penalty function diversity measure.
- (2) DIVACE-II PFC: Explorative version with PFC diversity measure.
- (3) DIVACE(R)-II: Restrictive version with NCL penalty function diversity measure.

Table 3

Confidence intervals with a confidence level of 95% for training and testing of DIVACE-II and other algorithms on both datasets. Results computed using accuracy rates obtained from 10- and 12-folds for the Australian and Diabetes datasets, respectively

Algorithm	Training		Testing	
	Australian	Diabetes	Australian	Diabetes
DIVACE-II	0.877 ± 0.005	0.771 ± 0.004	0.895 ± 0.0223	0.789 ± 0.0146
DIVACE-II PFC	0.876 ± 0.003	0.776 ± 0.003	0.889 ± 0.0159	0.785 ± 0.0222
DIVACE(R)-II	0.880 ± 0.003	0.764 ± 0.002	0.884 ± 0.0158	0.770 ± 0.0192
DIVACE(R)-II PFC	0.877 ± 0.003	0.763 ± 0.003	0.894 ± 0.0159	0.772 ± 0.0239
MPANN1	0.854 ± 0.011	0.771 ± 0.013	0.862 ± 0.0303	0.779 ± 0.0186
MPANN2	0.852 ± 0.009	0.755 ± 0.011	0.844 ± 0.0347	0.744 ± 0.0192
DIVACE	0.867 ± 0.004	0.783 ± 0.003	0.857 ± 0.0303	0.766 ± 0.0322
EENCL	0.891 ± 0.006	0.802 ± 0.004	0.857 ± 0.0241	0.764 ± 0.0237
Average variability	± 0.0075	± 0.007	± 0.0298	± 0.0234

(4) DIVACE(R)-II PFC: Restrictive version with PFC diversity measure.

We compare all these versions with MPANN (both variants from [3]—we refer to these as MPANN1 and MPANN2 here), DIVACE [16] and EENCL [35] due to the experimental setup⁸ in all these being similar and a direct comparison with other evolutionary ensemble construction methods being difficult [35]. We do compare our approach with 21 other learning algorithms from the literature as well. A comparison with non-evolutionary methods for ensemble construction viz. Bagging [9], Arcing [10] and AdaBoost [24], is also considered.

Table 3 shows interesting properties of the algorithms (all versions) in that, the mean test accuracy is higher than the mean training accuracy for both datasets, which mainly suggests that (on an average) the generalisation ability of DIVACE-II is good, i.e. it does not seem to overfit. MPANN2, DIVACE and EENCL on the other hand have higher mean training accuracies and so it can be said that, although these methods hold promise and do show good signs of generalisation, DIVACE-II performs even better due to its test accuracy being much higher. MPANN1 is the other algorithm having a mean test accuracy greater than its mean training accuracy but here again, the mean test accuracy (and mean training accuracy) is not better than DIVACE-II (except for the case of DIVACE(R)-II and DIVACE(R)-II PFC for the Diabetes dataset which are still better in both training and testing accuracies than MPANN2, DIVACE and EENCL).

⁸*n*-fold cross validation used here. *n* = 10 for Australian and *n* = 12 for Diabetes dataset. Learning rate for NNs is not the same as that used in [3,16,35] as the evolutionary process is inherently very different and we use methodologically different learners. Moreover, we evolve the population for 50 generations as opposed to 200 in [3,16,35] because the results (as will be seen later) were better than the compared evolutionary approaches, making further evolution unessential. The nature of the algorithm, whether or not increasing the number of generations would result in overfitting, would be investigated as part of future work.

Taking the case of training for both datasets (from Table 3), it can be seen that the four versions of DIVACE-II have very low variance as far as the acceptable values of the mean training accuracies are concerned. As compared to ± 0.005 , ± 0.003 , ± 0.003 and ± 0.003 for all four versions of DIVACE-II, other approaches show slightly more variable characteristics in having confidence intervals of ± 0.011 , ± 0.009 , ± 0.004 and ± 0.006 , respectively. On an average (averaging out the confidence intervals established to illustrate the difference and calling the result as ‘average variability’), the four approaches considered for comparison have confidence intervals of the order ± 0.0075 whereas for all versions of DIVACE-II this is ± 0.0035 .

A similar situation can be seen for the Diabetes dataset where we have ± 0.003 for all four versions of DIVACE-II as opposed to ± 0.007 for others. So, we can say that DIVACE-II is less variable, i.e. performs well on the training front.

On the testing front, for the Australian credit dataset, the confidence intervals established for DIVACE-II (all versions) can be given by ± 0.0174 whereas these are ± 0.0298 for other approaches. The latter is higher, signifying more variability on an average. Same is true for the Diabetes dataset where the intervals established by DIVACE-II are of the order of ± 0.0199 as opposed to ± 0.0234 for others. Generally speaking, DIVACE-II does compare well with previously studied approaches. The stability (low values for standard deviation) of DIVACE-II (explorative replacement version with NCL penalty function diversity measure) over multiple repetitions of cross validation is depicted in Table 4.

Tables 5 and 6 put DIVACE-II (all four versions considered) against 25 previously studied learning algorithms. The tables shows the average test error rates⁹ (lower means better) for many learning algorithms together with those for DIVACE-II.

⁹The two rates for DIVACE are from the use of different diversity measures (original formulation and PFC, respectively).

Table 4
Average performance (training and testing accuracy rates) of DIVACE-II on both datasets. Results averaged on 10 cross validation repetitions

	Australian		Diabetes	
	Training	Testing	Training	Testing
Mean (SD)	0.875 (0.003)	0.897 (0.005)	0.768 (0.004)	0.781 (0.009)

Table 5
Comparison of DIVACE-II with other learning algorithms from [37] in terms of the average test error rates for the Australian credit card assessment dataset. Results are averaged on 10-fold cross validation

Algorithm	Error rate	Algorithm	Error rate
DIVACE-II	0.105	CART	0.145
DIVACE-II	0.111	IndCART	0.152
PFC			
DIVACE(R)-II	0.116	NewID	0.181
DIVACE(R)-II	0.106	AC^2	0.181
PFC			
DIVACE	0.138, 0.125	Baytree	0.171
MPANN1	0.135	NaiveBay	0.151
MPANN2	0.156	CN2	0.204
EENCL	0.135	C4.5	0.155
Discrim	0.141	ITrule	0.137
Quadisc	0.207	Cal5	0.131
Logdisc	0.141	DIPOL92	0.141
SMART	0.158	BP	0.154
ALLOC80	0.201	RBF	0.145
k-NN	0.181	LVQ	0.197
CASTLE	0.148		

The algorithms used for comparison can be categorised into five classes: evolutionary ensemble methods (DIVACE, MPANN [3], EENCL [35]), statistical methods (Discrim, Quadisc, Logdisc, SMART, ALLOC80, k-NN, CASTLE, NaiveBay), decision trees based methods (CART, IndCART, NewID, AC^2 , Baytree, Cal5, C4.5), rule based methods (CN2, ITrule) and neural network based methods (BP, LVQ, RBF, DIPOL92). Details of the algorithms in the latter four classes can be found in [37]. The error rates refer to the percentage of wrong classifications on the test set.

As is evident from the tables, DIVACE-II (and the various versions thereof) has been able to achieve a good generalisation performance. It is generally better than other algorithms shown for both datasets, which shows the promise of not only DIVACE-II but also the evolutionary hybrid ensemble construction framework described in the paper. Moreover, since it is based on DIVACE and has outperformed it on both datasets considered here, the idea of enforcing diversity at multiple levels under a multi-objective evolutionary setup to find a good trade-off between diversity and accuracy seems to be of great value.

Tables 7 and 8 give the error rates of three non-evolutionary ensemble learning algorithms viz. Bagging,

Table 6
Comparison of DIVACE-II with other learning algorithms from [37] in terms of the average test error rates for the Diabetes dataset. Results are averaged on 12-fold cross validation

Algorithm	Error rate	Algorithm	Error rate
DIVACE-II	0.211	CART	0.255
DIVACE-II	0.215	IndCART	0.271
PFC			
DIVACE(R)-II	0.230	NewID	0.289
DIVACE(R)-II	0.228	AC^2	0.276
PFC			
DIVACE	0.227, 0.226	Baytree	0.271
MPANN1	0.221	NaiveBay	0.262
MPANN2	0.254	CN2	0.289
EENCL	0.221	C4.5	0.27
Discrim	0.225	ITrule	0.245
Quadisc	0.262	Cal5	0.25
Logdisc	0.223	DIPOL92	0.224
SMART	0.232	BP	0.248
ALLOC80	0.301	RBF	0.243
k-NN	0.324	LVQ	0.272
CASTLE	0.258		

Table 7
Results from some non-evolutionary ensemble learning algorithms in terms of the average test error rates for the Australian credit card assessment dataset. Results are averaged on 10-fold cross validation

Algorithm	Error rate
Bagging	0.138
Arcing	0.158
AdaBoost	0.157

Arcing and AdaBoost, for the two datasets. For the Australian credit card assessment dataset, all versions of DIVACE-II outperform the non-evolutionary methods. In case of the Diabetes dataset, DIVACE-II is essentially better except for in the restrictive case, where it is still comparable to Bagging and better on the other two.

7. Conclusion and directions for further research

All ensemble learning methods are essentially based on a very simple idea and they strive to achieve this goal: the goal of having diverse and accurate members within them which help the ensemble to outperform single learners. Recently, we also proposed an algorithm called DIVACE [16,18], the idea behind which was to enforce diversity and accuracy within the ensemble explicitly within a multi-objective evolutionary setup. This was found to be promising and so we tried to incorporate as much information (as far as diversity enforcement in ensembles is concerned) as possible into it so as to develop a framework which may be used as an ensemble learning algorithm generating engine.

This resulted in an evolutionary framework that uses a myriad of diversity enforcement ideas rolled into one

Table 8

Results from some non-evolutionary ensemble learning algorithms in terms of the average test error rates for the Diabetes dataset. Results are averaged on 12-fold cross validation

Algorithm	Error rate
Bagging	0.228
Arcing	0.244
AdaBoost	0.233

multi-level ensemble learning strategy where individual predictors are generated automatically by successively competing and co-operating with each other.

We also presented an algorithm, called DIVACE-II, which tries to model the framework proposed. This was used to prove the effectiveness/validity of the framework. Detailed experiments with this resulting algorithm showed that the framework is indeed valid. DIVACE-II was able to outperform most of the algorithms it was compared with. This indicates that our idea of enforcing diversity at multiple levels (which is modeled by our framework) is a good one. The framework can be used to generate diverse hybrid ensembles that generalise well.

Some more work needs to be done on the proposed algorithm. Firstly, evaluation of DIVACE-II on more and larger datasets will be carried out. We need to do more experiments with the version of the algorithm which models all three levels of evolution (from the framework) to truly understand the co-evolutionary ideas behind the framework. Level 3 evolution needs to be incorporated with all subsets/species evolving simultaneously. Also, other ways of population initialisation (and offspring generation at the training set level) need to be experimented with, e.g. input decimation [41] or feature selection together with bagging and *BagBoost*.

The methods explicated on in this paper need further analysis, both empirically and theoretically, in order to establish this framework as a truly generic model for ensemble algorithm synthesis. Some of the ideas and a summary of results expressed in this paper have been submitted to be included in a book chapter [15].

Acknowledgements

This research was undertaken as part of the EPSRC funded project on Market-Based Control of Complex Computational Systems (GR/T10671/01). This is a collaborative project involving the Universities of Birmingham, Liverpool and Southampton and BAE Systems, BT and HP.

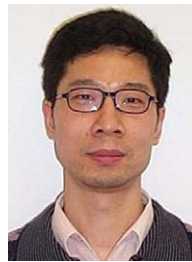
References

- [1] H.A. Abbass, A memetic pareto evolutionary approach to artificial neural networks, in: Proceedings of the 14th Australian Joint Conference on Artificial Intelligence, Springer, Berlin, 2000, pp. 1–12.
- [2] H.A. Abbass, Pareto neuro-ensemble, in: Proceedings of the 16th Australian Joint Conference on Artificial Intelligence, Perth, Australia, Springer, Berlin, 2003, pp. 554–566.
- [3] H.A. Abbass, Pareto neuro-evolution: constructing ensemble of neural networks using multi-objective optimization, in: The IEEE 2003 Conference on Evolutionary Computation, vol. 3, IEEE Press, New York, 2003, pp. 2074–2080.
- [4] H.A. Abbass, Speeding up backpropagation using multiobjective evolutionary algorithms, *Neural Comput.* 15 (11) (2003) 2705–2726.
- [5] H.A. Abbass, R. Sarker, C. Newton, Pde: a pareto-frontier differential evolution approach for multi-objective optimization problems, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC2001), vol. 2, IEEE Press, New York, 2001, pp. 971–978.
- [6] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: bagging, boosting, and variants, *Mach. Learning* 36 (1–2) (1999) 105–139.
- [7] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, 1995.
- [8] E. Boers, M. Borst, I. Sprinkhuizen-Kuyper, Evolving artificial neural networks using the “baldwin effect”, Technical Report 95-14, Leiden University, Department of Computer Science, The Netherlands, 1995.
- [9] L. Breiman, Bagging predictors, *Mach. Learning* 24 (2) (1996) 123–140.
- [10] L. Breiman, Bias, variance, and arcing classifiers, Technical Report 460, Statistics Department, University of California, Berkeley, 1996.
- [11] G. Brown, Diversity in neural network ensembles, Ph.D. Thesis, School of Computer Science, University of Birmingham, 2004.
- [12] G. Brown, J. Wyatt, R. Harris, X. Yao, Diversity creation methods: a survey and categorisation, *J. Inf. Fusion* 6 (2005) 5–20 (special issue on diversity in multiple classifier systems).
- [13] A. Chandra, Evolutionary approach to tackling the trade-off between diversity and accuracy in neural network ensembles, Technical Report, School of Computer Science, The University of Birmingham, UK, April 2004.
- [14] A. Chandra, Evolutionary framework for the creation of diverse hybrid ensembles for better generalisation, Master's Thesis, School of Computer Science, The University of Birmingham, Birmingham, UK, September 2004.
- [15] A. Chandra, H. Chen, X. Yao, Multi-objective machine learning, Trade-off between diversity and accuracy in ensemble generation, *Computational Intelligence*, Springer, Berlin, 2005, to appear (Chapter).
- [16] A. Chandra, X. Yao, DIVACE: diverse and accurate ensemble learning algorithm, in: Proceedings of the Fifth International Conference on Intelligent Data Engineering and Automated Learning, Exeter, UK, August 2004, Lecture Notes in Computer Science, vol. 3177, Springer, Berlin, pp. 619–625.
- [17] A. Chandra, X. Yao, Evolutionary framework for the construction of diverse hybrid ensembles, in: M. Verleysen (Ed.), Proceedings of the 13th European Symposium on Artificial Neural Networks, Brugge, Belgium, April 2005, pp. 253–258.
- [18] A. Chandra, X. Yao, Ensemble learning using multi-objective evolutionary algorithms, *J. Math. Modelling Algorithms* (2006), to appear.
- [19] P.J. Darwen, X. Yao, Every niching method has its niche: fitness sharing and implicit sharing compared, in: Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature (PPSN-IV), Lecture Notes in Computer Science, vol. 1141, Springer, Berlin, September 1996, pp. 398–407.
- [20] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley, Chichester, UK, 2001.
- [21] T.G. Dietterich, Machine-learning research: four current directions, *AI Mag.* 18 (4) (1998) 97–136.
- [22] V. Faber, Clustering and the continuous k -means algorithm, in: *Los Alamos Science: High Performance Computing*, vol. 22, Los Alamos National Laboratory, 1994, pp. 138–144.

- [23] S. Forrest, R.E. Smith, B. Javornik, A.S. Perelson, Using genetic algorithms to explore pattern recognition in the immune system, *Evol. Comput.* 1 (3) (1993) 191–211.
- [24] Y. Freund, R. Schapire, A short introduction to boosting, *J. Jpn. Soc. Artif. Intell.* 14 (5) (1999) 771–780.
- [25] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: *Proceedings of the 13th International Conference on Machine Learning*, Morgan Kaufmann, Los Altos, CA, 1996, pp. 148–156.
- [26] S. Geman, E. Bienenstock, R. Doursat, Neural networks and the bias/variance dilemma, *Neural Comput.* 4 (1) (1992) 1–5.
- [27] S. Gutta, J. Huang, I.F. Imam, H. Wechsler, Face and hand gesture recognition using hybrid classifiers, in: *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition (FG '96)*, IEEE Computer Society, 1996, pp. 164–170.
- [28] L.K. Hansen, P. Salamon, Neural network ensembles, *IEEE Trans. Pattern Anal. Mach. Intell.* 12 (10) (1990) 993–1001.
- [29] T.K. Ho, J.J. Hull, S.N. Srihari, Decision combination in multiple classifier systems, *IEEE Trans. Pattern Anal. Mach. Intell.* 16 (1) (1994) 66–75.
- [30] Y. Jin, B. Sendhoff, Reducing fitness evaluations using clustering techniques and neural networks ensembles, in: *Genetic and Evolutionary Computation Conference*, Seattle, USA, Springer, Berlin, July 2004, pp. 688–699.
- [31] K. Kottathra, Y. Attikiouzel, A novel multicriteria optimization algorithm for the structure determination of multilayer feedforward neural networks, *J. Network Comput. Appl.* 19 (1996) 135–147.
- [32] A. Krogh, J. Vedelsby, Neural network ensembles, cross validation, and active learning, *Neural Inf. Process. Syst.* 7 (1995) 231–238.
- [33] W.B. Langdon, S.J. Barrett, B.F. Buxton, Combining decision trees and neural networks for drug discovery, in: *Genetic Programming, Proceedings of the Fifth European Conference, EuroGP 2002*, Kinsale, Ireland, 3–5 April 2002, pp. 60–70.
- [34] Y. Liu, X. Yao, Ensemble learning via negative correlation, *Neural Networks* 12 (10) (1999) 1399–1404.
- [35] Y. Liu, X. Yao, T. Higuchi, Evolutionary ensembles with negative correlation learning, *IEEE Trans. Evol. Comput.* 4 (4) (2000) 380.
- [36] R. Meir, G. Raetsch, An introduction to boosting and leveraging, *Advanced Lectures on Machine Learning*, Springer-Verlag New York, Inc. 2003, pp. 118–183.
- [37] D. Michie, D. Spiegelhalter, C. Taylor, *Machine Learning Neural and Statistical Classification*, Ellis Horwood, Chichester, UK, 1994.
- [38] D. Opitz, R. Maclin, Popular ensemble methods: an empirical study, *J. Artif. Intell. Res.* 11 (1999) 169–198.
- [39] D.W. Opitz, J.W. Shavlik, Generating accurate and diverse members of a neural-network ensemble, *Neural Inf. Process. Syst.* 8 (1996) 535–541.
- [40] N.C. Oza, Online ensemble learning, Ph.D. Thesis, Computer Science Division, University of California, Berkeley, 2001.
- [41] N.C. Oza, K. Tumer, Input decimation ensembles: decorrelation through dimensionality reduction, in: *Proceedings of the International Workshop on Multiple Classifier Systems*, Cambridge, UK, June 2001, *Lecture Notes in Computer Science*, vol. 2096, Springer, Berlin, pp. 238–247.
- [42] A. Sharkey, Multi-Net Systems, *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, Springer, Berlin, 1999, pp. 1–30 (Chapter).
- [43] A. Sharkey, N. Sharkey, Combining diverse neural networks, *Knowl. Eng. Rev.* 12 (3) (1997) 231–247.
- [44] A.J.C. Sharkey, Types of multinet system, in: *Proceedings of the International Workshop on Multiple Classifier Systems*, Cagliari, Italy, June 2002, *Lecture Notes in Computer Science*, vol. 2364, Springer, Berlin, pp. 108–117.
- [45] K.O. Stanley, R. Miikkulainen, Evolving neural networks through augmenting topologies, *Evol. Comput.* 10 (2) (2002) 99–127.
- [46] K. Tumer, J. Ghosh, Analysis of decision boundaries in linearly combined neural classifiers, *Pattern Recognition* 29 (2) (1996) 341–348.
- [47] N. Ueda, R. Nakano, Generalization error of ensemble estimators, in: *Proceedings of the International Conference on Neural Networks*, 1996, pp. 90–95.
- [48] G. Valentini, F. Masulli, Ensembles of learning machines, in: R. Tagliaferri, M. Marinaro (Eds.), *Neural Nets WIRN Vietri-2002*, *Lecture Notes in Computer Science*, vol. 2486, Springer, Berlin, June 2002, pp. 3–19.
- [49] W. Wang, P. Jones, D. Partridge, Diversity between neural networks and decision trees for building multiple classifier systems, in: *Proceedings of the International Workshop on Multiple Classifier Systems*, Cagliari, Italy, June 2000, *Lecture Notes in Computer Science*, vol. 1857, Springer, Berlin, pp. 240–249.
- [50] W. Wang, D. Partridge, J. Etherington, Hybrid ensembles and coincident-failure diversity, in: *Proceedings of the International Joint Conference on Neural Networks*, Washington, USA, July 2001, vol. 4, IEEE Press, New York, pp. 2376–2381.
- [51] K. Woods, W. Kegelmeyer, K. Bowyer, Combination of multiple classifiers using local accuracy estimates, *IEEE Trans. Pattern Anal. Mach. Intell.* 19 (1997) 405–410.
- [52] X. Yao, Evolving artificial neural networks, in: *Proceedings of IEEE*, vol. 87, IEEE Press, New York, 1999, pp. 1423–1447.
- [53] W. Yates, D. Partridge, Use of methodological diversity to improve neural network generalization, *Neural Comput. Appl.* 4 (2) (1996) 114–128.



Arjun Chandra is a Ph.D. student at the School of Computer Science, University of Birmingham, UK. He received his M.Sc. degree in Natural Computation from the University of Birmingham, UK, in December 2004 and his B.Tech. in Computer Science and Engineering from Dr. Ram Manohar Lohia Avadh University, Faizabad, India, in 2002. His research interests include multi-objective evolutionary algorithms, co-evolutionary learning, ensemble learning, probabilistic modeling and game theory.



Xin Yao received the B.Sc. degree from the University of Science and Technology of China (USTC), Hefei, the M.Sc. degree from the North China Institute of Computing Technologies (NCI), Beijing, and the Ph.D. degree from the USTC, in 1982, 1985, and 1990, respectively, all in computer science.

He is currently a Professor of Computer Science and the Director of the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), University of Birmingham, U.K. He is also a Distinguished Visiting Professor of USTC and a Cheung Scholar awarded by the Ministry of Education of China. He was a Lecturer, Senior Lecturer, and an Associate Professor at University College, University of New South Wales, the Australian Defence Force Academy (ADFA), Canberra, Australia, between 1992–1999. He held Postdoctoral Fellowships from the Australian National University (ANU), Canberra, and the Commonwealth Scientific and Industrial Research Organization (CSIRO), Melbourne, between 1990 and 1992. His major research interests include evolutionary computation, neural network ensembles, global optimization, computational time complexity, and data mining.

He is an IEEE Fellow, the Editor-in-Chief of *IEEE Transactions on Evolutionary Computation* and the recipient of the 2001 IEEE Donald G. Fink Prize Paper Award. He has given more than 35 invited keynote and plenary speeches at various conferences worldwide. He has more than 200 publications in evolutionary computation and computational intelligence.