Adaptive Evolutionary Artificial Neural Networks for Pattern Classification

Tatt Hee Oong and Nor Ashidi Mat Isa, *Member, IEEE*

***Abstract*— This paper presents a new evolutionary approach called the hybrid evolutionary artificial neural network (HEANN) for simultaneously evolving an artificial neural networks (ANNs) topology and weights. Evolutionary algorithms (EAs) with strong global search capabilities are likely to provide the most promising region. However, they are less efficient in fine-tuning the search space locally. HEANN emphasizes the balancing of the global search and local search for the evolutionary process by adapting the mutation probability and the step size of the weight perturbation. This is distinguishable from most previous studies that incorporate EA to search for network topology and gradient learning for weight updating. Four benchmark functions were used to test the evolutionary framework of HEANN. In addition, HEANN was tested on seven classification benchmark problems from the UCI machine learning repository. Experimental results show the superior performance of HEANN in fine-tuning the network complexity within a small number of generations while preserving the generalization capability compared with other algorithms.**

***Index Terms*— Adaptive evolution, neural network design, pattern classification.**

I. Introduction

ARTIFICIAL neural networks (ANNs) have emerged as a powerful tool for pattern classification [1], [2]. The optimization of ANN topology and connection weights training are often treated separately. Such a divide-and-conquer approach gives rise to an imprecise evaluation of the selected topology of ANNs. In fact, these two tasks are interdependent and should be addressed simultaneously to achieve optimum results.

One of the key tasks of pattern classification is designing a compact and well-generalized ANN topology. Choosing an appropriate ANN topology for specific problems is critical for ANN generalization because of the strong correlation between the information processing capability and the ANN topology. An excessively small network size suggests that the problem cannot be learned well, whereas an excessively large network size will lead to over-fitting and poor generalization performance. Time-consuming trial-and-error approaches and hill-climbing constructive or pruning algorithms [3]–[7] used to design an ANN architecture for a given task only explore small architectural subsets and tend to be stopped at structural local optima. The cascaded-correlation neural network [8] is a popular constructive algorithm used to construct ANN topologies that have multiple layers. New hidden nodes are added one by one and are connected with every existing hidden node in the current network. Thus, the network can be seen as having multiple one-unit layers that form a cascade structure. However, the network is prone to structural local optima because of its constructive behavior. Designing an ANN topology using evolutionary algorithms (EAs) has become a popular method to overcome the drawbacks of the constructive or pruning approaches [9]–[13]. EAs, which have a strong global search capability, can effectively search through the near-complete class of ANN topologies.

Much work has been devoted to the evolution of ANN topologies. Two major approaches to evolving ANN topologies reported in the literature are the evolution of ANN topology without weights and the simultaneous evolution of both topology and weights. For the evolution of an ANN topology without weights, the ANN topology has to be trained from a random set of initial weights to evaluate its fitness. Yao and Liu [11] noted that this fitness evaluation method is very noisy because a phenotype's fitness is used to represent the genotype's fitness. Although the fitness of the evolved ANN topology can be estimated by using average results of multiple runs from different sets of random initial weights, the computational time for fitness evaluation is increased dramatically. Thus, only small ANN topologies are evolved in [14] and [15].

For the simultaneous evolution of both ANN topology and weights, the information about ANN topology and weight is encoded in each individual. Thus, the noisy fitness evaluation problem can be alleviated. One prominent work called EPNet and introduced by Yao and Liu [11] is an example of the simultaneous evolution of both ANN topology and weights. Yao and Liu used hybrid training to evolve the ANN weights. Gradient learning and simulated annealing are both incorporated into the evolutionary process to evolve the ANN weights. Similarly, Martínez-Estudillo *et al*. [16] used a gradient learning method to evolve the weights of the best individual ANN in each cluster of the population. They used a clustering algorithm to partition the individuals in the population into several clusters that belong to the same region of attraction. However, the use of a gradient learning method such as a backpropagation algorithm to evolve ANN weights

Manuscript received December 20, 2010; revised September 15, 2011; accepted September 15, 2011. Date of publication October 3, 2011; date of current version November 2, 2011. This work was supported in part by Universiti Sains Malaysia under the Postgraduate Fellowship Scheme and the Research University Grant, under the project Study on Capability of Fourier Transform Infrared Spectral Characteristic for Development of Intelligent Cervical Precancerous Diagnostic System.

The authors are with the Imaging and Intelligent Systems Research Team, School of Electrical and Electronic Engineering, Universiti Sains Malaysia, Nibong Tebal 14300, Malaysia (e-mail: othee@hotmail.com; ashidi@eng.usm.my).

Digital Object Identifier 10.1109/TNN.2011.2169426

causes noisy fitness evaluation because of the high sensitivity of the backpropagation algorithm to the initial weights. Palmes *et al*. [12] used mutation-based EA to address this issue in an evolutionary ANN. Other works have also focused on the simultaneous evolution of ANN topology and weights. Ang *et al*. [13] introduced growth probability to allow the network to evolve to a suitable size. Angeline *et al*. [9] used parametric mutations and structural mutations to evolve ANN weights, hidden nodes and networks links. Jian and Yugeng [10] used evolutionary programming (EP) to evolve the architecture and weights of feedforward neural networks and recurrent neural network. Leung *et al*. [17] proposed an improved genetic algorithm to evolve the structure and parameters of neural networks. Gutiérrez *et al*. [18] used simulated annealing to control the parametric mutation and five structural mutations to evolve the radial basis function (RBF) neural networks. All of these previous works were intended to produce a compact and well generalized ANN.

Typically, the topology of an ANN can be encoded into a chromosome using a direct encoding scheme or an indirect encoding scheme. In the direct encoding scheme, all of the ANN topology is encoded directly into a chromosome by using a binary representation that indicates the existence of network connections and hidden nodes. The indirect encoding scheme only encodes some important parameters of an ANN topology, such as the number of hidden layers and hidden nodes. Other details about the ANN topology are predefined or specified by a set of deterministic developmental rules [19]. The direct encoding scheme is easy to implement and suitable for the precise and fine-tuned search. Although the indirect encoding scheme can reduce the length of chromosome, it may not be appropriate for finding a compact ANN with good generalization ability [19]. In this paper, a direct encoding scheme is used to represent the ANN topology.

Evolution of the ANN topology and weights can help to alleviate the problem of noisy evaluation. However, relying solely on the EA to evolve the ANNs is rather inefficient in performing a local search. Further training using conventional gradient descent methods after the evolution can be a simple approach to fine-tune a network. Considered the situation of premature convergence in the evolutionary process, the ANN found after further training may not be optimum. Other attempts such as the use of the annealing temperature to guide the weight perturbation step size [9] and scheduling of the mutation probability [12] of an individual network in the population can be considered preliminary guides toward a local search, but no distinct direction has been provided. Hence, the EA should be guided properly to maintain the balance between the exploration of the entire search space and the exploitation of important regions.

Motivated by the importance of balancing the global and local search in the evolutionary process and concurrent evolution of the ANN topology and weights, this paper presents a new evolutionary approach called the hybrid evolutionary artificial neural network (HEANN) that simultaneously evolves ANN topology and weights. Moreover, HEANN demonstrates a balance between global search and local search by introducing a novel adaptive mutation technique in the evolutionary process. Instead of using the conventional schedule method to reduce the mutation probability or step size over time, HEANN uses the information from the population to guide the evolutionary process to shift gradually from a global search to local search. First, the generalization loss (*GL*) in the population is used to adapt the mutation probability and step size of the entire population. Second, the fitness value of each individual in the population is used to determine the severity of the mutation in a given individual in the population.

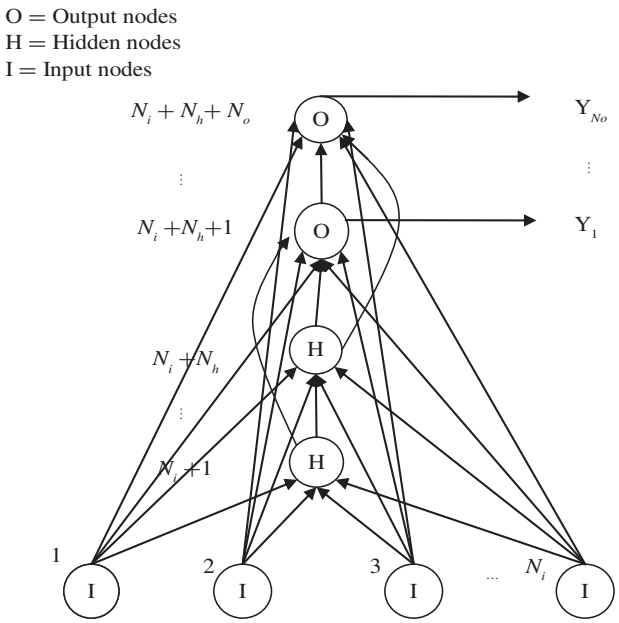

Fig. 1. Arbitrarily connected feedforward neural network.

In this paper, there are two main factors to be optimized: the network topology and the generalization capability. A popular method for optimizing two or more objectives is based on Pareto dominance [20]–[22]. However, the computational cost for finding and estimating the fitness of each Pareto front is increased when there is an increase in the number of objectives to be optimized [21]. Another popular approach is based on scalarized multiobjective learning [10], [12], [20], which aggregates several objectives into a scalar cost function. In this case, the approach assumes the convexity of the Pareto frontier. This approach is used in current implementation.

The rest of this paper is organized as follows. Section II describes HEANN and each component in detail. Section III analyzes the effects of the proposed adaptive mutation technique on the ANN topology (adding or deleting nodes, connections) and how this technique balances the global and local search of the evolutionary process. Section IV shows the results of this new adaptive evolutionary framework for four benchmark functions. Section V contains the experimental results of HEANN. Section VI presents the discussion of results. Finally, Section VII concludes the work of this paper.

II. HEANN

It is crucial to choose appropriate search operators when evolving ANNs. Crossover- and mutation-based EAs are two

TABLE I
DESCRIPTION OF HEANN ADAPTIVE MUTATION PARAMETERS

Notation(s)	Description
P_s	Structural mutation probability
	This parameter determines the addition and deletion of hidden nodes and network connections
P_w	Weight mutation probability
	This parameter determines whether the connection weights should be mutated, i.e., adding random Gaussian noise with zero mean and step size of SS
SS	Step size of weight perturbation
P_{gs}, P_{gw}, and SS_g	Global structural mutation probability, global weight mutation probability, and global step size of weight perturbation
	These parameters determine the mutability of the entire population
P_{ls}, P_{lw}, and SS_l	Local structural mutation probability, local weight mutation probability, and local step size of weight perturbation
	These parameters determine the mutability of an individual in the population
$tmpP_{gs}$, $tmpP_{gw}$, and $tmpSS_g$	Temporary global structural mutation probability, temporary global weight mutation probability, and temporary global step size of weight perturbation
	These parameters determine the decay rate of P_{gs}, P_{gw}, and SS_g
$baseP_{ls}$, $baseP_{lw}$, and $baseSS_l$	Base local structural mutation probability, base local weight mutation probability, and base local step size of weight perturbation
	These parameters determine the maximum achievable value of P_{ls}, P_{lw}, and SS_l

popular approaches used to evolve ANNs. ANNs utilize distributed representations in which the knowledge is distributed among all of the weights in an ANN. The use of crossover to recombine one part of an ANN with another part of an ANN is likely to destroy both ANNs [19]. However, crossover is useful for an ANN that utilizes a local representation such as the RBF. In addition, Spears and Anand [23] claim that crossover is effective in the evolution of ANNs when the population size is sufficiently large, but no general guideline is provided for how large a population size should be to improve the result. Nevertheless, the permutation problem stands as a major issue in crossover-based evolutionary ANNs [19]. Thus, EP [24], [25] is adopted in HEANN and emphasizes the preservation of the behavioral links between parent and child. Hence, no recombination operator is used in HEANN. Moreover, HEANN introduces a novel adaptive mutation strategy to enhance the balance between the global and local search capabilities of the evolutionary process. Although HEANN is used to evolve feedforward ANNs with sigmoid activation functions (Fig. 1) in this paper, it is not restricted to the use of a specific classifier. Rather, the evolutionary technique used in HEANN can be applied to any type of ANN with minimal constraint.

A. Overview

In this sub-section, an overview of the HEANN algorithm will be presented through a description of adaptive mutation strategy and a step-by-step illustration. Detailed descriptions about each component of HEANN including the encoding scheme, fitness function, stopping criteria, and selection mechanism are given in the subsequent sub-sections.

The balance between global search and local search of the EA is dictated by the values of mutation probability and perturbation step size. Large values of mutation probability and perturbation step size boost the exploration at the expense of exploitation. Although a conventional scheduling method can reduce the mutation probability or step size over time, it is difficult to determine the decay rate. Rather, HEANN uses two main guidelines to adapt the mutation probability and step size of the evolutionary process. First, the GL in the population is used to adapt the mutation probability and step size of the entire population. Second, the fitness value of each individual in the population is used to determine the severity of the mutation in a given individual in the population. The GL is described in detail in Section II-C. Table I shows the general descriptions of HEANN adaptive mutation parameters. Guided by the GL in the population, the global structural mutation probability (P_{gs}), global weight mutation probability (P_{gw}) and global step size of the weight perturbation (SS_g) at generation t are defined as follows:

$$P_{gs}(t) = tmpP_{gs} * \left(\frac{s(t)}{S}\right) \tag{1}$$

$$P_{gw}(t) = tmpP_{gw} * \left(\frac{s(t)}{S}\right) \tag{2}$$

$$SS_g(t) = tmpSS_g * \left(\frac{s(t)}{S}\right) \tag{3}$$

where $tmpP_{gs}$, $tmpP_{gw}$, and $tmpSS_g$ are updated with $P_{gs}(t)$, $P_{gw}(t)$, and $SS_g(t)$, respectively, when $GL(t) < 0$. S is the maximum consecutive generation allowable for $GL \geq 0$, $s(t)$ is the current value of the stopping counter. The values of $tmpP_{gs}$, $tmpP_{gw}$, and $tmpSS_g$ are initialized with the user-specified value. Typically, the initial value of $tmpP_{gs}$ and $tmpP_{gw}$ can be set to slightly higher than the conventional mutation probability (0.01) because P_{gs} and P_{gw} can be

adapted during evolutionary process. The value of $tmpSS_g$ is initialized to a small value (≤ 5) such that the sigmoid function of the ANN will not enter the saturation region easily. An analysis of the effect of mutation parameters is presented in Section III.

From (1)–(3), it is clear that $tmpP_{gs}$, $tmpP_{gw}$, and $tmpSS_g$ determine the decay rates of P_{gs}, P_{gw}, and SS_g. As the stopping counter decreases throughout the evolutionary process, P_{gs}, P_{gw}, and SS_g also decrease slowly. Consequently, HEANN shifts gradually from a global search to a local search. Every time a better solution is found ($GL(t) < 0$), it is assumed that the evolutionary process has yet to converge to global minimum (or even a local minimum). Thus, $tmpP_{gs}$, $tmpP_{gw}$, and $tmpSS_g$ are updated so that the decay rates of P_{gs}, P_{gw}, and SS_g are reduced to account for more time in local search.

In addition to adapting the mutability of the entire population, HEANN adapts the mutation probability for a given individual in the population. Individual mutation operations are guided by fitness values. The local structural mutation probability (P_{ls}), local weight mutation probability (P_{lw}) and local step size of weight perturbation (SS_l) of the n-th individual in the population are defined as follows:

$$P_{ls}(n) = baseP_{ls} * \left(1 - \frac{F_{\text{best}}}{F(n)}\right) \tag{4}$$

$$P_{lw}(n) = baseP_{lw} * \left(1 - \frac{F_{\text{best}}}{F(n)}\right) \tag{5}$$

$$SS_l(n) = baseSS_l * \left(1 - \frac{F_{\text{best}}}{F(n)}\right) \tag{6}$$

where F_{best} is the lowest (best) fitness value of a network in the population in the current generation, $F(n)$ is the fitness value of n-th individual in the population; and $baseP_{ls}$, $baseP_{lw}$, and $baseSS_l$ are the user-specified base local structural mutation probability, base local weight mutation probability, and base local step size of weight perturbation, respectively, for an individual. Typically, the value of $baseP_{ls}$ and $baseP_{lw}$ can be set to slightly higher than the conventional mutation probability (0.01), and the value of $baseSS_l$ is set to a small value (≤ 5) such that the sigmoid function of the ANN does not enter saturation region easily.

It is observed that in (4)–(6), the values of P_{ls}, P_{lw}, and SS_l are not greater than $baseP_{ls}$, $baseP_{lw}$, and $baseSS_l$. A low-fitness (better) individual will have lower values of P_{ls}, P_{lw}, and SS_l, whereas high-fitness individuals will have higher P_{ls}, P_{lw}, and SS_l to preserve the good individuals within the population. Note that the values of P_{ls}, P_{lw}, and SS_l are zero for the solution with best fitness, which is acceptable because the mutation operation depends on both global and local mutation strategies as described below.

The structural mutation probability (P_s), weight mutation probability (P_w), and step size of the weight perturbation (SS) of the n-th individual at generation t are the sum of their global and local components defined as follows:

$$P_s(n,t) = P_{gs}(t) + P_{ls}(n) \tag{7}$$

$$P_w(n,t) = P_{gw}(t) + P_{lw}(n) \tag{8}$$

$$SS(n,t) = SS_g(t) + SS_l(n). \tag{9}$$

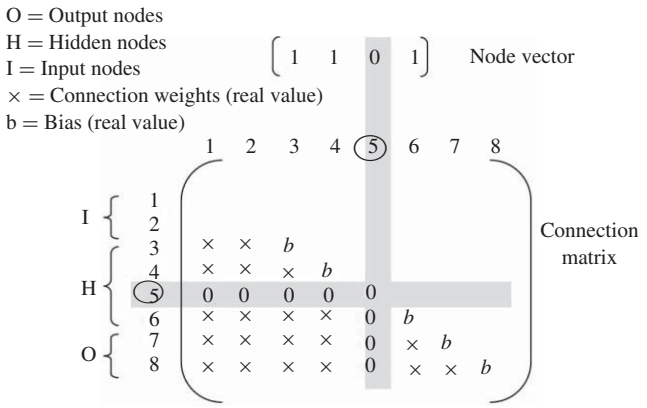

Fig. 2. Node vector and connection matrix of ANN. A zero entry in the node vector indicates that this particular node does not exist, and all connections from and to this node are discarded (highlighted row and column).

A detailed analysis about global mutation strategy and local mutation strategy is presented in Section III.

The major steps of HEANN are explained as follows.

Step 1: Generate an initial population of μ ANNs with random hidden nodes and random connections. Assign the initial weights including the network bias with uniform random values between -1 and 1. Initialize S, $baseP_{ls}$, $baseP_{lw}$, $baseSS_l$, $tmpP_{gs}$, $tmpP_{gw}$, and $tmpSS_g$.

Step 2: Calculate the local structural mutation probability (P_{ls}), local weight mutation probability (P_{lw}), and local step size of weight perturbation (SS_l) of each individual network in the population according to (4)–(6), respectively. P_{ls}, P_{lw}, and SS_l are used to determine the severity of the mutation in a given individual in the population.

Step 3: Calculate $P_{gs}(t)$, $P_{gw}(t)$, and $SS_g(t)$ at generation t according to (1)–(3), respectively. $P_{gs}(t)$, $P_{gw}(t)$, and $SS_g(t)$ are used to guide the entire population's exploration and exploitation of the search space. Note that this step is executed at intervals of generations (10 generations are used in the current implementation) instead of every generation.

Step 4: Perform the structural mutation (adding or deleting hidden nodes and network connections) and weight mutation (Gaussian perturbation of weights) on the node vector and connection weight matrix (Fig. 2) for each individual in the parent population. Hidden nodes of an ANN are mutated by bit flipping in the node vector. Connection deletion is performed by defining the non-zero value in the connection matrix as zero. Conversely, connection addition is achieved by initiating a zero value in the connection matrix with uniform random values between -1 and 1. The probability of structural mutation is based on (7). Next, the probability that a non-zero connection weight of a network will be mutated is based on (8). If successful, this particular connection weight is mutated by adding a Gaussian perturbation with mean 0 and a step size as described in (9), which is defined as follows:

$$w_{ij} = w'_{ij} + N\left(0, \left(SS_l + SS_g\right)\right) \tag{10}$$

where w_{ij} and w'_{ij} denote the parent and offspring connection weight values in the connection matrix. $N(0,\sigma)$ is the

Gaussian perturbation with mean 0 and standard deviation σ. In this case, $\sigma = SS_l + SS_g$.

Step 5: Evaluate each offspring created after the mutation according to the classification error in the validation set. If the best network found in the current generation is better than the reserved best network (ANN_{best}) up to this generation, replace ANN_{best} with this network and reset the stopping counter to its initial value, S. Under the same condition, update the temporary global structural mutation probability ($tmpP_{gs}$), temporary global weight mutation probability ($tmpP_{gw}$), and temporary global step size of weight perturbation ($tmpSS_g$) to $P_{gs}(t)$, $P_g(t)_w$, and $SS_g(t)$, respectively. If there is a tie in the classification error at the validation set when finding ANN_{best}, the individual with the lowest classification error at the training set will be the new ANN_{best}. If a tie still exists, the individual with the fewest connections and input nodes will be the new ANN_{best}. Decrease the stopping counter by one and do not update $tmpP_{gs}$, $tmpP_{gw}$, and $tmpSS_g$ if the classification error at the validation set of the best network found in the current generation is equal to or greater than ANN_{best}.

Step 6: Stop the evolutionary process and go to Step 8 if the stopping counter is decreased to zero or the maximum number of generations has been reached. Otherwise, proceed to the next step.

Step 7: Sort the offspring according to their fitness values and use the rank-based selection to choose μ–2 offspring to become parents for the next generation. Elitism is also used in HEANN, where one fittest parent and one fittest offspring from the current generation will be retained for the next generation. Hence, μ ANNs are selected as parents for the next generation. Steps 2–7 are repeated until the termination criterion for stopping the evolutionary process is satisfied.

Step 8: The best network in terms of the classification error in the validation set (ANN_{best}) is the final ANN for the given problem.

B. Encoding Scheme

In this paper, a HEANN is used to evolve arbitrarily connected feedforward neural networks (ACN) (Fig. 1), where there is a minimum constraint for the connections between neurons. A direct connection from input nodes to output nodes and a connection across layers are allowed. ACN has been proven to be more robust than the standard multilayer perceptron (MLP) and requires fewer neurons to fulfill the same task [26], [27]. It is also clear that the standard MLP is a subset of ACN. At one extreme, a fully connected feedforward neural network can be formed when all possible feedforward connections are connected in the ACN.

In total, $N_i + N_h + N_o$ nodes are present in the network, where N_i, N_h, and N_o are the numbers of input, hidden, and output nodes, respectively. The maximum number of hidden nodes allowable in the ANN is a user-specified parameter, whereas the number of input nodes and output nodes is problem-dependent. The direct encoding scheme is used to represent the network topology and the connection weights, including biases (Fig. 2). A N_h dimensional vector represents the hidden nodes of a particular ANN with binary entries (i.e., hidden nodes for this given example), which indicates the existence of these nodes (1 = exist; 0 = does not exist). For the network connections and weights, a $(N_i + N_h + N_o) \times (N_i + N_h + N_o)$ matrix is sufficient to represent all possible connections and weights in the ACN (Fig. 2) (i.e., two output nodes for this given example). The entries of the connection matrix are real numbers representing weights ($\times$ in Fig. 2) and biases (b in Fig. 2) of a network. A zero entry indicates that no connection exists between two nodes indexed by the row and column numbers of the matrix.

Note that the connection matrix representing the ANN structure will be triangular below the diagonal because no feedback connection from a neuron with high indices to one with a lower index is allowed. In addition, a zero entry in the node vector indicates that this particular node does not exist, and all connections to and from this node are discarded from the connection matrix (highlighted row and column in Fig. 2), which shows that the hidden nodes of an ANN are evolvable by simply flipping a bit in the node vector. The connection direction is defined from the node with column number i to the node with row number j in the connection matrix. The connection weights are updated by Gaussian perturbation in the mutation operation as described in Section II-A, Step 4.

C. Fitness Function and Stopping Criteria

The fitness function (F) used in HEANN is given by a weighted sum of two important factors: training error (φ_t) and network complexity (φ_c)

$$F = \alpha_1 \varphi_t + \alpha_2 \varphi_c \tag{11}$$

where α_1 and α_2 are user-defined small-value parameters between 0 and 1. The training error (φ_t) and network complexity (φ_c) are defined as

$$\varphi_t = \frac{100}{np} \sum_{t=1}^{p} \sum_{i=1}^{n} (T_i(t) - Y_i(t))^2 \tag{12}$$

$$\varphi_c = \ln C \tag{13}$$

where n and p are the number of output nodes and number of training patterns, respectively; $T_i(t)$ and $Y_i(t)$ represent the target and actual outputs of the ith output nodes for a training pattern t; C denotes the number of connections in the network; logarithm scaling is used in (13) to avoid large values as C increases. Note that the lower fitness value implies a fitter individual.

The network topology and weights can grow indefinitely when direct encoding is used to represent an ANN. Although the maximum number of hidden nodes allowable in the ANN is specified by the user, the network size can grow to its maximum bound. Thus, the use of a fitness function to penalize the large network is another positive feature to obtain compact network topology.

The values of α_1 and α_2 are used to control the contribution of their respective factors in the overall fitness value. Typically, a high contribution of network training error in the overall fitness value and a small penalty toward network complexity

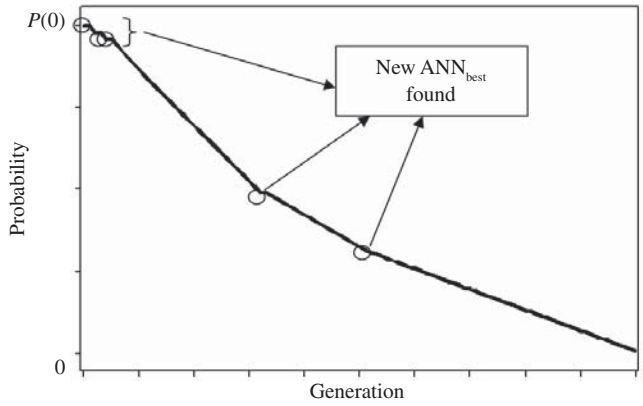

Fig. 3. Illustration of global adaptive mutation strategy.

are preferable for evolving compact ANNs with a good generalization capability. The values of $\alpha_1 = 1$ and $\alpha_2 = 0.1$ are used based on the results of preliminary experiments.

Stopping the evolutionary process after an appropriate number of generations is a crucial step to ensure that the evolved ANN does not lose its generalization capability or suffer from premature convergence. In HEANN, classification error at the validation set is used to monitor the *GL* in the population. According to Prechelt [28], the *GL* at generation t can be defined as follows:

$$GL(t) = \left(\frac{E_{\text{best}}(t) + 1}{E_{\text{opt}}(t) + 1} - 1 \right) \tag{14}$$

where $E_{\text{best}}(t)$ is the lowest classification error for the validation set found in the population of offspring at generation t; $E_{\text{opt}}(t)$ is the lowest classification error for the validation set found up to generation t. A high *GL* can be an obvious reason for stopping the training. However, in early generations, the training performance and validation performance are both inconsistent, and deciding to stop the evolutionary process based solely on the *GL* is unreliable and may lead to premature convergence. Hence, HEANN allows the evolutionary process to continue for an extra S generations before stopping when there is a consecutive *GL* or no change in the best accuracy, i.e., $GL \geq 0$ consecutively for S generation.

There are two conclusions that can be drawn from $GL \geq 0$ in the population. First, over-fitness is apparent in the population. Second, the population has located a good region in the search space but faces difficulty in local search. A stopping counter with the initial value of S is used to flag a stopping signal when it decreases to zero. The stopping counter will be decreased by one in the particular generation if $GL \geq 0$ and will be reset to its initial value S if an ANN with a better classification accuracy at the validation set than ANN_{best} is found, i.e., $GL < 0$. These extra generations are necessary to avoid over-fitness or premature convergence. The best ANN with the lowest classification error at the validation set (ANN_{best}) found up to the current generation is the final ANN for the given problem. If there is a tie in the classification error for the validation set when finding ANN_{best}, the individual with the lower classification error for the training set will be selected as the final result. If a tie still exists, the individual with fewer connections and input nodes will be selected as the final result. The final ANN will then be tested on an unseen testing set to evaluate its performance. The stopping criteria are similar to the sliding window of Palmes *et al*. [12] except that the *GL* is observed in every generation to prevent good ANN results among the internal windows from being eliminated.

D. Selection Mechanism

To maintain the population diversity at each generation and prevent super individuals from dominating the whole population [29], HEANN uses rank-based selection [30] to choose μ–2 offspring to become parents for the next generation. Moreover, elitism [29] is used in HEANN where one fittest parent and another fittest offspring from the current generation is also retained for the next generation. Thus, μ ANNs are selected as parents for the next generation. Each individual in the parent population undergoes a structural mutation (adding or deleting nodes and connections) and weight mutation (Gaussian perturbation of weights).

III. Analysis of Adaptive Mutation Strategy

The values of P_s, P_w, and *SS* are often fixed in the conventional evolutionary ANNs. However, biological evolution shows that these values are dependent on the evolution state and should be adapted [31]. This section analyzes and further explains the relationship between the global adaptive strategy and local adaptive strategy of HEANN.

A. Global Adaptive Mutation Strategy

Fig. 3 shows the strategy for adapting the global mutation probability (P_g) of HEANN. Rather than scheduling the P_g over time until the maximum generation is reached or the evolutionary process satisfies the termination criterion, HEANN uses the *GL* from the entire population to determine the decay rate of P_g. In the initial stages of the evolutionary process where ANN candidates are scattered around the search space, the P_g of the entire population is high, and there is a high probability of finding a new ANN_{best}. Thus, P_g hardly decreases to allow the exploration of better search regions.

In the intermediate stage (or pre-maturing stage) of the evolutionary process, the use of a large P_g is no longer feasible to find a better ANN_{best}, which also contributes to the long execution and poor convergence of the evolutionary process. Hence, P_g decays when there is no new ANN_{best} found in the population indicated by $GL \geq 0$ consecutively. As P_g decays, HEANN gains a better local search capability. A new ANN_{best} found in the intermediate stage of the evolutionary stage will reduce the decay rate of P_g with the aim of extending the local search period and avoiding over-fitness or premature convergence.

Finally, when the low P_g can no longer provide a high probability of finding a better ANN_{best}, i.e., no ANN_{best} is found for S consecutive generations, HEANN is converged and the evolutionary process will be terminated. Here, HEANN escapes from local optima by the rank-based selection mechanism. Low-performance ANNs located far from the current

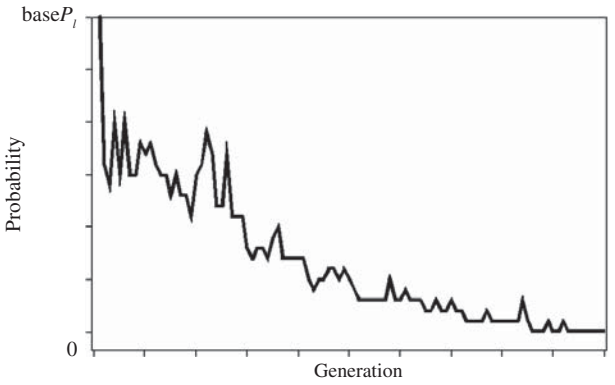

Fig. 4. Variation of average local mutation probability.

important region will not necessarily be eliminated from the population, and HEANN will continue to look for a better solution. Thus, HEANN can still explore new regions of the search space because of the high local mutation probability, even though the evolutionary process has reached its final stage. The analysis of the global adaptive mutation probability is valid for SS_g because the same adaptive strategy is being used.

B. Local Adaptive Mutation Strategy

Unlike the global adaptive mutation strategy, each individual in the population has a different local mutation probability (P_l). Fig. 4 depicts the general trend of the average local mutation probability in the population for a typical case (obtained from the results of a diabetes problem). P_l fluctuates wildly in the initial stages of the evolutionary process because of the scattered solutions and rank-based selection mechanism of HEANN. Moreover, a large difference between $F(n)$ of the n-th individual in the population and F_{best} accounts for the large average mutation probability in the initial stage of the evolutionary process. Although P_l fluctuates throughout the evolutionary process, it exhibits a decreasing trend in average value, which indicates that the $F(n)$ of n-th individual in the population is close to F_{best}. In other words, HEANN converges to an optimum solution.

To prevent the disruption of the near-optimal ANNs or premature convergence, the HEANN local adaptive mutation strategy assigns a lower value of P_l for high-fitness ANNs and a higher value of P_l for low-fitness ANNs. This is achieved by (4)–(6). Coupled with the rank-based selection mechanism, the low-fitness ANNs prevent HEANN from being stranded at a local optimum, whereas the high-fitness ANNs aid in the convergence of HEANN. This analysis is applicable to SS_l because the same adaptive strategy is being used.

C. Combination of Global and Local Adaptive Mutation Strategies

The global adaptive mutation strategy controls the mutability of the entire population, and the local adaptive mutation strategy determines the severity of the mutation for a given individual in the population. Thus, HEANN emphasizes the exploration of the entire search space and the exploitation of important regions. In the initial stage of the evolutionary process, the value of P_g and the average value of P_l are sufficiently large to account for the exploration of the entire search space. As P_g decreases, the evolutionary process enters the intermediate stage, where high-fitness ANNs start to fine-tune the network parameters and connections, meanwhile, low-fitness ANNs continue to search for better search regions. Lastly, the low P_g in the final stage of the evolutionary process emphasizes the exploitation of important regions. Only a small portion of the population with large P_l remains for exploration. As a result, HEANN overcomes the shortcoming of EAs in fine-tuning ANNs while preserving the global search capability of EAs.

IV. Benchmark Functions

The evolutionary framework of HEANN as described above is designed to evolve ANN. To examine the performance of this evolutionary framework in optimizing the local search capability of the evolutionary process and gaining a greater probability of escaping from local minima, four benchmark functions [17], [32] are used to test the evolutionary framework. We call this evolutionary framework a hybrid EA (HEA). The details of the HEA are the same as those described in Section II except for the weight perturbation step size, which is equivalent to the input variable perturbation step size in the benchmark functions, and the weight mutation probability, which is equivalent to the input variable mutation probability. The structural mutation probability is ignored in this test.

The four functions $f_i(\mathbf{x}); i = 1, 2, 3, 4$, where $\mathbf{x} = [x_1, x_2, \ldots, x_{30}]^T$ is a 30-D input, are defined as follows.

1) *Sphere Model*

$$f_1(\mathbf{x}) = \sum_{i=1}^{30} x_i^2, \qquad -5.12 \le x_i \le 5.12$$
$$\min(f_1) = f_1(0, \ldots, 0) = 0. \tag{15}$$

2) *Schwefel's Problem* 2.22

$$f_2(\mathbf{x}) = \sum_{i=1}^{30} |x_i| + \prod_{i=1}^{30} |x_i|, \qquad -10 \le x_i \le 10$$
$$\min(f_2) = f_2(0, \ldots, 0) = 0. \tag{16}$$

3) *Generalized Rastrigin's Function*

$$f_3(\mathbf{x}) = \sum_{i=1}^{30} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right],$$
$$-5.12 \le x_i \le 5.12$$
$$\min(f_3) = f_3(0, \ldots, 0) = 0. \tag{17}$$

4) *Generalized Griewank Function*

$$f_4(\mathbf{x}) = \frac{1}{4000} \sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{30} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1,$$
$$-600 \le x_i \le 600$$
$$\min(f_4) = f_4(0, \ldots, 0) = 0 \tag{18}$$

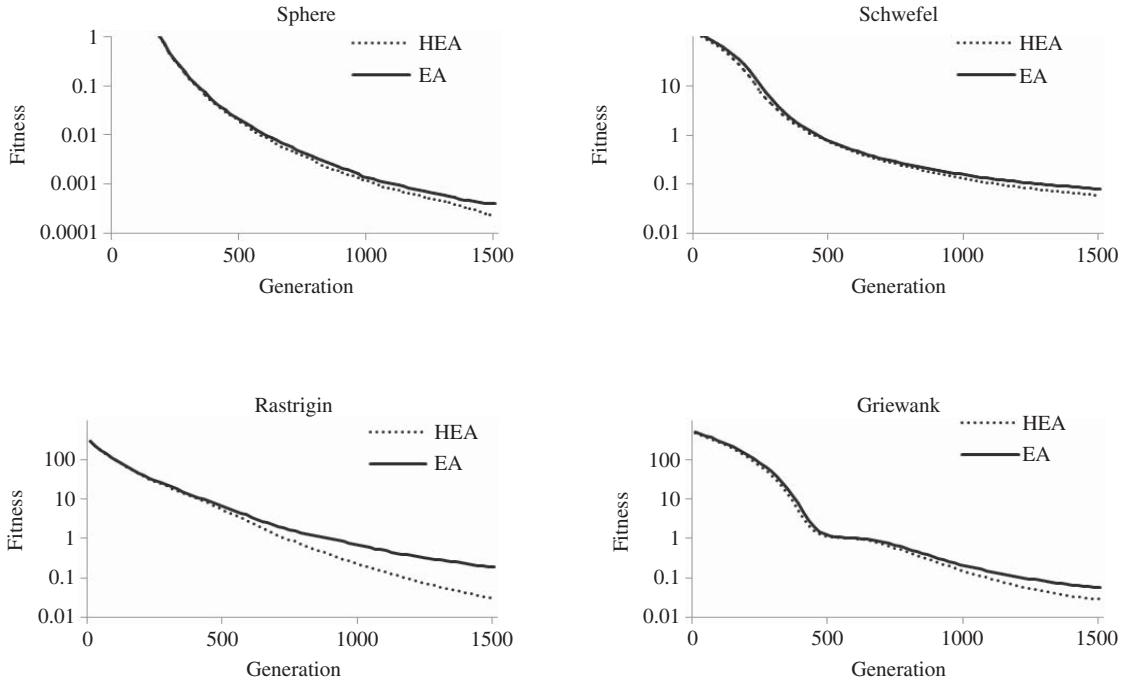

Fig. 5. Comparison between HEA and EA on functions f_1 to f_4.

TABLE II
COMPARISON BETWEEN HEA AND EA ON FOUR BENCHMARK FUNCTIONS

Function	HEA		EA		t-test against HEA
	Mean	Std. Dev.	Mean	Std. Dev.	
f_1	2.40×10^{-4}	8.30×10^{-5}	3.91×10^{-4}	9.10×10^{-5}	1.09×10^{-13}
f_2	5.97×10^{-2}	9.31×10^{-3}	8.00×10^{-2}	1.47×10^{-2}	2.14×10^{12}
f_3	3.05×10^{-2}	1.01×10^{-2}	1.91×10^{-1}	7.12×10^{-2}	3.25×10^{-21}
f_4	2.91×10^{-2}	5.57×10^{-3}	5.69×10^{-2}	1.33×10^{-2}	7.01×10^{-9}

where functions f_1 and f_2 are unimodal, and functions f_3 and f_4 are multimodal. The number of local minima for functions f_3 and f_4 increases exponentially with the problem dimension [32]. Functions f_1 and f_2 are aimed to investigate the strength of HEA to finely tune the local minima. Functions f_3 and f_4 are used to examine the ability of HEA to escape from poor local minima and finely tune a good search region.

The results are then compared with those obtained from conventional EA, which has a constant mutation probability and mutation step size. For each test function, the population size is 100 and the test is run over 1500 generations. These parameters are the same for HEA and EA. The $baseP_{lw}$ and $tmpP_{gw}$ for HEA assume the same value to maintain the balance between global search and local search. These values for functions f_1 to f_4 are set to 0.01, 0.01, 0.02, and 0.02, respectively. Similarly, the $baseSS_l$ and $tmpSS_g$ assume the same value and are set to 1, 1, 1, and 20 for the four benchmark functions, respectively. The mutation probabilities of EA for functions f_1 to f_4 are set to 0.01, 0.01, 0.02, and 0.02, respectively, and the mutation perturbation step sizes are set to 1, 1, 1, and 20, respectively. These values are selected from preliminary experiments for good performance. The initial population is generated uniformly at random in the range specified in (15)–(18).

Fig. 5 shows the progress of the mean best solutions found by HEA and EA over 50 runs for functions f_1 to f_4. The convergence rate of HEA and EA are almost equal during the early stage of the evolution, but HEA outperforms EA in the final stage of the evolution. Moreover, the average results of 50 independent runs shown in Table II indicate that HEA outperforms EA with a 95% confidence level. Functions f_1 and f_2 are unimodal, and the better results of HEA over EA in these functions indicate that EA is weaker than HEA in fine tuning. Such weakness is apparent when the solution is near the global minimum. The ability of HEA to escape from local minima can be observed from the results for functions f_3 and f_4. HEA performs significantly better than EA consistently for these two functions.

V. RESULTS OF EVOLVING ANN

To examine the performance of HEANN, seven benchmark classification problems were selected from the UCI machine learning repository [33] and are summarized in Table III. The chosen datasets cover examples of different difficulties

TABLE III
CHARACTERISTICS OF SEVEN BENCHMARK PROBLEMS

Data set	Inputs	Classes	Size of training set	Size of validation set	Size of testing set
Cancer	9	2	349	175	175
Diabetes	8	2	384	192	192
Iris	4	3	75	38	37
Wine	13	3	89	45	44
Mammography	5	2	481	240	240
Blood	4	2	374	187	187
Survival	3	2	154	76	76

and various sizes of input attributes, output classes, and data patterns.

A. Experimental Setup

Prechelt [28] proposed a set of ANN benchmarking methodologies. We followed these methodologies to set up the experiments for all problems to allow a fair comparison with other similar works. The datasets of all problems were partitioned into three sets: a training set, a validation set and a testing set following the partition rule of Prechelt [28]. The number of examples in these sets is shown in Table III. The input attributes of all problems were rescaled to between 0 and 1 by a linear function. The output attributes of all problems were encoded using a 1-of-m output representation for the m classes. The output node with the highest activation indicates the output class (the winner-take-all strategy). The initial connection density of the ANNs was set to 0.75, and the initial connection weights were assigned uniform random values between -1 and 1.

The use of conventional classification accuracy measures will disproportionately emphasize large classes. Other generalization error measures such as the receiver-operating characteristic curve and area under the curve are useful for problems with unbalanced classes [34]. However, these two measures are only suitable for two class problems. As used in this paper, the average class error rate is independent of class distribution and is defined as follows:

$$\text{average class error rate} = \frac{1}{m}\sum_{i=1}^{m}\frac{e_i}{n_i} \tag{19}$$

where m is the number of classes in a problem, e_i is the number of misclassified examples of class i and n_i is the total number of example of class i.

The compactness of the ANN is measured by the number of connections of a network. Moreover, Akaike's information criterion (AIC) [35] is used to evaluate the structure of an ANN that combines the number of free parameters in an ANN and the output error of the ANN as an estimated statistical model. The AIC penalizes the number of parameters to discourage overfitting. The corrected version of AIC used in this paper is defined as follows:

$$\text{AIC} = \frac{n\ln(\text{MSE}) + 2K + (2K(K+1))}{(n-K-1)} \tag{20}$$

where n is the sample data size, MSE is the mean square error of ANN, and K is the number of estimated parameters in ANN (i.e., number of connections in the ANN). A lower AIC indicates better ANN structure.

To further justify the importance of combining the local and global adaptive mutations (HEANN-GL), HEANN with a global adaptive mutation (HEANN-G) and HEANN with a local adaptive mutation (HEANN-L) were compared in the experiments. In addition, ANNs evolved from the normal EA with fixed mutation probability (EANN) were also compared. About 50 independent runs were simulated for each variant of HEANN and EANN. Some control parameters need to be set. However, HEANN is not very sensitive to these parameters, and tuning all of them is unnecessary. Table IV lists the implementation details of three HEANN variants and EANN used in the experiments. EANN has a fixed mutation probability of 0.01 and weight perturbation step size of 1. For HEANN variants, the mutation probability and weight perturbation step size decayed from 0.02 and 2, respectively, (global) or fluctuated up to a maximum of 0.02 and 2, respectively (local). Furthermore, a few similar works were chosen for comparison with HEANN. This paper does not seek to compare HEANN exhaustively with all other similar algorithms because it is difficult to compare the designs of the ANNs with EA. Therefore, the comparisons here mainly serve as a guide to evaluate the performance of HEANN for different problems. In addition, HEANN is also compared with other leading classification techniques used for modern classification benchmarks, i.e., backpropagation-trained ANN (BP-ANN), support vector machine (SVM), k-nearest neighbor (k-NN) algorithm and C4.5 decision tree.

B. Experimental Results

The results presented in Table V show that HEANN-GL achieves the lowest test error rate (TER) in all problems. The TER is tabulated using (19). Although the t-test based on the TER shows a close match among HEANN variants and EANN in four out of seven problems, the network complexity and the number of generations of HEANN-GL are significantly lower than those observed for HEANN-L and EANN at a 95% confidence level, with the exception of the iris problem, in which the number of generations of HEANN-GL is significantly lower than that of HEANN-L and EANN at 90% confidence level. Furthermore, HEANN-GL has the lowest AIC value in all problems, which indicates that HEANN-GL evolves a better ANN structure than all other algorithms. HEANN-GL performs better than HEANN-G in terms of network complexity in six out of seven problems. The number of generations of HEANN-GL and HEANN-G does not show a significant difference. In addition to the comparison between HEANN variants and EANN, the results of the EPNet [11], mutation-based genetic neural network (MGNN-ep) [12], and self-adaptive growth-probability-based neural network (NN-SAGP) [13] are also compared. EPNet uses five mutation operators, which include gradient learning to modify the ANN architecture and weights. MGNN-ep uses the EP and scheduled mutation probability for weight learning. NN-SAGP

TABLE IV
IMPLEMENTATION DETAILS

Features	Algorithms			
	HEANN-GL	HEANN-G	HEANN-L	EANN
Maximum local mutation probability and step size ($baseP_{ls}$, $baseP_{lw}$, $baseSS_l$)	0.02	–	0.02	–
	0.02		0.02	
	2		2	
Initial global mutation probability and step size [$P_{gs}(0)$, $P_{gw}(0)$, $SS_g(0)$]	0.02	0.02	–	–
	0.02	0.02		
	2	2		
Structural mutation probability (P_s)	$P_{ls} + P_{gs}$	P_{gs}	$0.01 + P_{ls}$	0.01
Weight mutation probability (P_w)	$P_{lw} + P_{gw}$	P_{gw}	$0.01 + P_{lw}$	0.01
Weight perturbation step size (SS)	$SS_l + SS_w$	SS_w	$1 + SS_l$	1
Stopping counter (S)	500			
Population size	100			
Maximum generation	3000			
Max hidden nodes	10			
Fitness constant	$\alpha_1 = 1$; $\alpha_2 = 0.1$; $\alpha_3 = 0$; ($\alpha_3 = 1$ for evolution includes input feature selection)			

uses the self-adaptive growth probability to evolve the weights and the hidden nodes of the ANNs. These algorithms share one ultimate goal, i.e., to design a compact and well-generalized ANN topology.

Table V shows that HEANN-GL outperforms MGNN-ep in terms of TER, the network complexity and the average number of generations in cancer, iris and wine problems. Although MGNN-ep uses a scheduled mutation probability to evolve ANNs, it only considers the severity of the mutation for a given individual in the population and neglects the adaptability of the entire population. This is the main reason for the difference in the performance of HEANN-GL and MGNN-ep. Compared with NN-SAGP, HEANN-GL has a lower TER and standard deviation in cancer and diabetes problems. The self-adaptive growth probability strategy of NN-SAGP is a good attempt at adapting the structural mutation of ANNs throughout the whole evolutionary process. However, NN-SAGP overlooks the importance of adapting the mutation probability, which plays a major role in parametric learning and fine tuning. Another comparison is performed between HEANN-GL and EPNet. The results in Table V show that HEANN-GL outperforms EPNet in terms of TER in cancer and diabetes problems. However, the network complexity is not significantly different between HEANN-GL and EPNet in cancer problems. This can be explained by the comparable performance between the adaptive mutation strategy of HEANN-GL and ordered mutation operators of EPNet, which encourage the parsimony of evolved ANNs. Nevertheless, EPNet still suffers from the noisy evaluation of its fitness function arising from its continuous reliance on gradient learning during mutation.

Table VI compares the HEANN results with those of other leading classification techniques on modern classification benchmarks, i.e., BP-ANN, SVM, k-NN algorithm and C4.5 decision tree. The parameters used by various classification techniques are also shown in this table. These parameters are selected by trial-and-error experiments for good performance. The results presented are the average of a single 10-fold cross-validation run. HEANN achieved the lowest TER in six out of seven problems. The TER measure is independent of the class distribution. Thus, it can be clearly seen that C4.5 decision tree perform poorly for imbalanced class distribution problems, such as diabetes, blood and survival problems. The SVM and k-NN algorithms achieve good results in non-noisy problems but perform poorly for noisy datasets such as diabetes and survival problems. The BP-ANN shows good average results in all problems. However, the performance of BP-ANN is not better than that of HEANN because the network structure obtained by trial and error might not be optimal.

C. Effect of Mutation Parameters

This section investigates the effect of different mutation parameters on the performance of HEANN and EANN. Three problems were selected for this analysis: cancer, diabetes and mammography problems. These problems were selected because they possess different class distributions and different difficulties. For example, the cancer problem is an easy problem and has an unbalanced class distribution, the diabetes problem is a difficult problem that has an unbalanced class distribution, the mammography problem possesses a moderate difficulty and a nearly balanced class distribution. For other control parameters, HEANN is not very sensitive to the distributions, and tuning them is unnecessary.

Fig. 6 compares the TER and AIC achieved by HEANN and EANN for cancer, diabetes and mammography problems

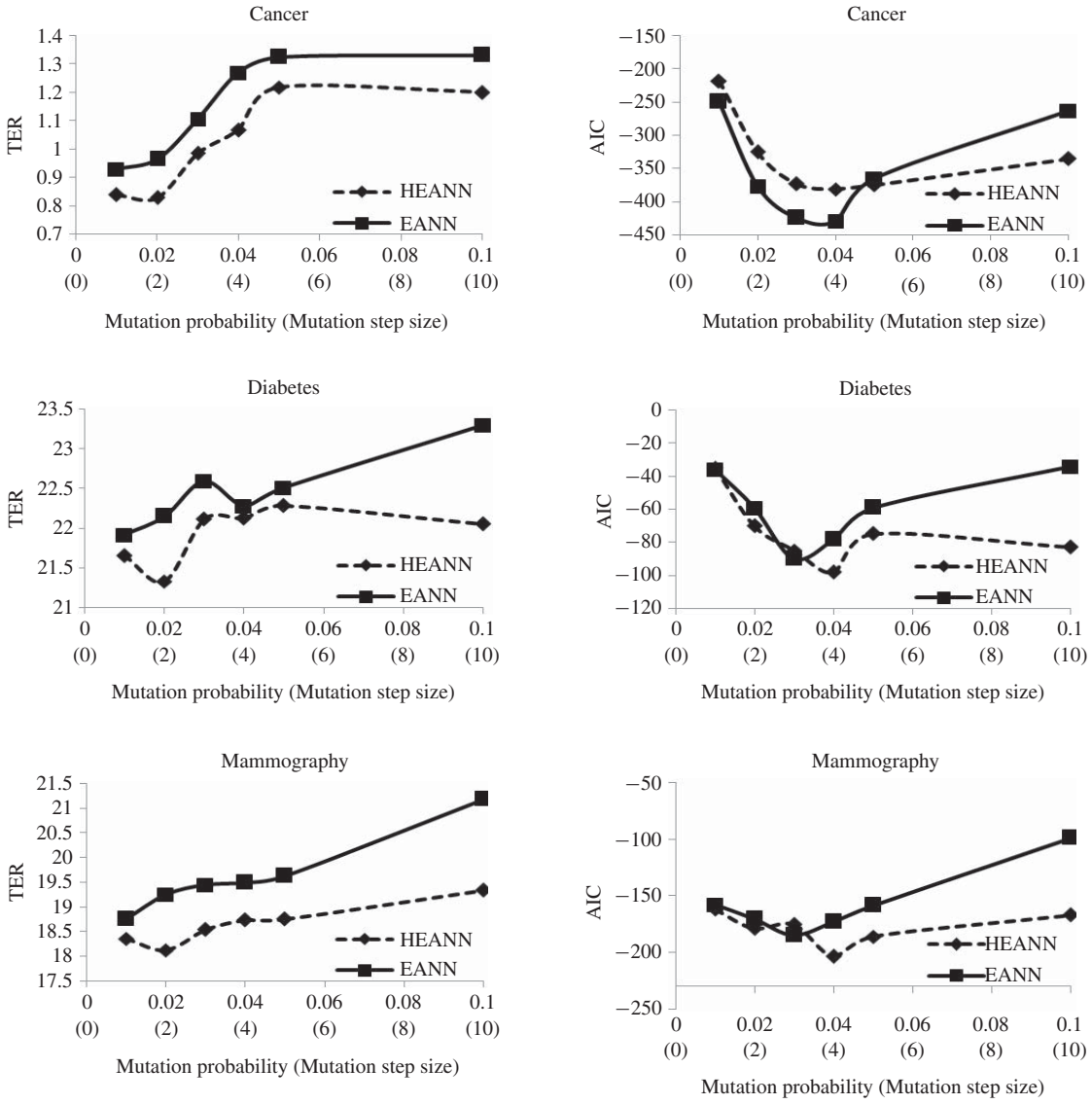

Fig. 6. Effect of mutation probability and mutation step size on HEANN and EANN.

using different mutation parameters. The values of P_s and P_w were varied from 0.01 to 0.10, and the value of SS was varied from 1 to 10. These parameters are proportional to each other because the increase in these parameters will favor a global search and achieve poor fine tuning. For HEANN, the $baseP_{ls}$, $tmpP_{gs}$, $baseP_{lw}$, and $tmpP_{gw}$ assume the same value to maintain the balance between global search and local search. These parameters are varied from 0.01 to 0.10. Similarly, the $baseSS_l$ and $tmpSS_g$ assumed the same value and were varied from 1 to 10. The results presented are the average of 50 independent runs. HEANN and EANN demonstrate an increasing trend with respect to TER when the mutation probability and mutation step size increase. However, HEANN is not very sensitive to the mutation parameters compared with EANN. Although a low mutation probability and mutation step size can produce ANN with low TER, the network structure evaluated using AIC is not optimal. Thus, the mutation probability and mutation step size should not be too low because the performance of EA will suffer from premature convergence.

VI. Discussion

The benchmark function test indicates that HEA has a better local search capability and a greater probability of escaping from local minima than conventional EA. HEA adaptively alters the mutation probability and mutation step size during the evolutionary process to consistently produce better results than EA, particularly in the final stage of the evolutionary process.

Table V suggests that HEANN with a better local search capability and a greater chance to escape from local minima also performs well in fine tuning ANN. Additionally, Table VI indicates that HEANN is more resistive to class-imbalanced problems such as cancer, diabetes, blood and survival problems. It is also worth noting that HEANN introduces little

TABLE V
ANN ARCHITECTURE DESIGN: COMPARISON RESULTS FOR SEVEN BENCHMARK CLASSIFICATION PROBLEMS

Algorithm	TER (Std. Dev.)	Connections (Std. Dev.)	Generations (Std. Dev.)	AIC	t-Test Against HEANN-GL		
					TER	Conn	Gen
Cancer							
HEANN-GL	**0.83 (0.50)**	48.48 (14.74)	78.42 (56.08)	**−325.43**	–	–	–
HEANN-G	0.92 (0.84)	52.74 (14.67)	**75.82 (61.75)**	–298.48	0.517	0.151	0.826
HEANN-L	0.89 (0.64)	60.34 (16.90)	129.44 (156.40)	–225.39	0.549	0.000	0.034
EANN	0.93 (0.53)	57.44 (15.80)	123.84 (114.00)	–249.36	0.333	0.004	0.014
MGNN-ep [11]	3.14	77.12	167.00	–	–	–	–
NN-SAGP [12]	1.65 (0.68)	–	–	–	–	–	–
EPNet [10]	1.38 (0.94)	**41.00 (14.70)**	–	–	–	–	–
Diabetes							
HEANN-GL	**21.33 (1.68)**	**47.56 (11.32)**	404.04 (143.60)	**−70.21**	–	–	–
HEANN-G	21.85 (1.54)	50.08 (12.53)	**374.58 (151.50)**	–63.38	0.110	0.294	0.321
HEANN-L	22.15 (1.49)	53.54 (11.32)	621.22 (388.00)	–51.57	0.011	0.010	0.000
EANN	21.91 (1.20)	57.68 (12.59)	547.54 (253.11)	–36.01	0.049	0.000	0.001
NN-SAGP [12]	24.16 (2.58)	–	–	–	–	–	–
EPNet [10]	22.38 (0.014)	52.30 (16.10)	–	–	–	–	–
Iris							
HEANN-GL	**0.70 (1.19)**	**31.30 (8.75)**	173.36 (176.20)	**−29.92**	–	–	–
HEANN-G	1.06 (1.53)	35.46 (9.22)	**131.98 (95.20)**	97.38	0.184	0.023	0.148
HEANN-L	0.81 (1.36)	38.10 (8.13)	378.80 (306.20)	–0.07	0.664	0.000	0.000
EANN	0.8 (1.65)	41.32 (9.71)	252.82 (230.20)	73.81	0.723	0.000	0.056
MGNN-ep [11]	4.68	53.52	378.60	–	–	–	–
Wine							
HEANN-GL	**3.06 (2.78)**	**55.42 (11.10)**	128.38 (84.10)	**−453.78**	–	–	–
HEANN-G	4.85 (2.99)	55.76 (9.34)	**116.62 (69.53)**	–385.24	0.002	0.869	0.448
HEANN-L	3.59 (2.66)	70.16 (9.90)	208.10 (132.3)	–371.85	0.325	0.000	0.001
EANN	4.63 (3.05)	68.92 (12.00)	221.38 (132.3)	–368.55	0.008	0.000	0.000
MGNN-ep [11]	4.68	129.70	619.20	–	–	–	–
Mammography							
HEANN-GL	**18.12 (1.11)**	42.86 (10.78)	**835.64 (296.10)**	**−178.91**	–	–	–
HEANN-G	18.32 (1.28)	**41.66 (11.34)**	875.06 (303.60)	–176.33	0.392	0.589	0.513
HEANN-L	18.87 (1.24)	49.44 (9.44)	1429.24 (474.90)	–154.98	0.002	0.002	0.000
EANN	18.76 (1.13)	51.50 (12.59)	1586.70 (520.20)	–158.01	0.005	0.000	0.000
Blood							
HEANN-GL	**31.84 (0.86)**	**22.04 (7.26)**	**203.18 (146.10)**	**−120.09**	–	–	–
HEANN-G	31.89 (1.19)	24.76 (9.35)	212.80 (189.10)	–111.64	0.808	0.108	0.777
HEANN-L	31.96 (1.68)	27.36 (11.97)	487.04 (418.30)	–110.44	0.668	0.009	0.000
EANN	32.00 (1.19)	25.38 (9.23)	447.62 (337.60)	–113.45	0.455	0.047	0.000
Survival							
HEANN-GL	**31.60 (2.54)**	**18.30 (6.42)**	**240.96 (146.90)**	**−1.10**	–	–	–
HEANN-G	32.12 (3.01)	21.30 (9.73)	276.80 (172.40)	16.72	0.358	0.072	0.266
HEANN-L	32.23 (2.79)	21.92 (8.65)	625.36 (358.50)	14.19	0.239	0.020	0.000
EANN	32.57 (2.63)	24.94 (11.98)	602.56 (410.80)	42.68	0.063	0.001	0.000

TER – Test Error Rate (%)
Conn – Connections
Gen – Generations
AIC – Akaike Information Criterion

overhead in computational time when adapting the mutation process compared with the fitness function. With the adaptive mutation strategy, HEANN achieves comparable or better results in fewer generations.

The structural mutation probability, weight mutation probability and step size of the weight perturbation are interdependent. For example, a small mutation probability should be coupled with a small step size for the weight perturbation

TABLE VI
COMPARISON BETWEEN HEANN AND BACKPROPOGATION-TRAINED ANN, SVM, k-NN AND C4.5 IN TERMS OF THE TESTING ERROR RATE FOR SEVEN BENCHMARK CLASSIFICATION PROBLEMS

Data Set	Test Error Rate (Std. Dev.) Parameters				
	HEANN	BP-ANN	SVM	k-NN	C4.5
Cancer	**0.83 (0.50)**	3.49 (2.55)	4.21 (3.71)	3.18 (3.41)	6.13 (4.41)
	Same with HEANN-GL	epoch = 100; $\alpha = 0.3$; $\beta = 0.5$	RBF kernel $C = 10$; $\gamma = 10$	$k = 5$	$m = 35$
Diabetes	**21.33 (1.68)**	25.87 (2.96)	30.82 (2.82)	29.06 (3.39)	37.42 (3.69)
	Same with HEANN-GL	epoch = 100; $\alpha = 0.3$; $\beta = 0.5$	RBF kernel $C = 40$; $\gamma = 30$	$k = 5$	$m = 230$
Iris	**0.70 (1.19)**	4.67 (4.50)	4.67 (3.22)	4.00 (3.44)	8.67 (3.22)
	Same with HEANN-GL	epoch = 10000; $\alpha = 0.3$; $\beta = 0.5$	RBF kernel $C = 10$; $\gamma = 10$	$k = 5$	$m = 60$
Wine	3.06 (2.78)	2.57 (3.56)	**1.90 (3.33)**	4.00 (3.86)	5.52 (5.81)
	Same with HEANN-GL	epoch = 10000; $\alpha = 0.3$; $\beta = 0.5$	RBF kernel $C = 10$; $\gamma = 10$	$k = 5$	$m = 1$
Mammography	**18.12 (1.11)**	19.13 (3.45)	20.84 (4.91)	20.53 (3.69)	35.51 (15.29)
	Same with HEANN-GL	epoch = 100; $\alpha = 0.3$; $\beta = 0.5$	RBF kernel $C = 10$; $\gamma = 10$	$k = 5$	$m = 192$
Blood	**31.84 (0.86)**	37.01 (5.60)	38.27 (4.14)	38.57 (5.00)	49.62 (1.20)
	Same with HEANN-GL	epoch = 10000; $\alpha = 0.3$; $\beta = 0.5$	RBF kernel $C = 50$; $\gamma = 500$	$k = 5$	$m = 7$
Survival	**31.60 (2.54)**	39.46 (10.80)	47.50 (6.83)	42.30 (7.07)	49.24 (4.03)
	Same with HEANN-GL	epoch = 10000; $\alpha = 0.3$; $\beta = 0.5$	RBF kernel $C = 50$; $\gamma = 50$	$k = 15$	$m = 3$

to account for the exploitation of important regions. Setting a small mutation probability with large mutation step size is not going to aid in exploitation because an individual will always take a large step in solution space. Moreover, the small mutation step size is used in HEANN to prevent the training process from entering the sigmoid saturation region, where the slope of sigmoid function is approaching zero.

VII. CONCLUSION

This paper described a new approach for simultaneously evolving ANN topology and weights. To address the weakness of EAs in finely tuning ANNs, HEANN uses an adaptive mutation strategy to refine the local search capability in the design of ANNs. *GL* and fitness value are considered to aid the mutation adaption in the evolutionary process. Furthermore, the global and local mutation strategies of HEANN are analytically studied.

HEA was first tested on four benchmark functions to demonstrate the improvement of local search capability and the ability to escape from local minima. The experimental results show that HEA performs better than conventional EA. Next, HEANN was tested on seven real-world classification problems. Overall, the results from experiments show the superiority of HEANN in various aspects compared with other algorithms. Very compact ANNs can be produced by HEANN with the adaptive mutation strategy. Moreover, HEANN requires fewer generations to achieve good results. The only drawback of HEANN is that it has many user-specified parameters. However, HEANN is not very sensitive to these parameters. The results of the different mutation parameters test show that HEANN is not very sensitive to these parameters, however, these values should not be too low.

Without the use of gradient learning, which requires intensive computation in the evolutionary process, the use of a larger population size is feasible in HEANN. The final population definitely contains more information than a single ANN. Thus, it would be challenging to formulate the adaptive mutation strategy to be suitable for evolving neural network ensembles in future work, particularly when several subpopulations are cooperating to solve a specific problem.

ACKNOWLEDGMENT

The authors are grateful to the anonymous reviewers for their constructive comments, which helped to improve this paper.

REFERENCES

[1] R. P. Lippmann, "Pattern classification using neural networks," *IEEE Commun. Mag.*, vol. 27, no. 11, pp. 47–50, Nov. 1989.
[2] G. P. Zhang, "Neural networks for classification: A survey," *IEEE Trans. Syst., Man, Cybern., Part C: Appl. Rev.*, vol. 30, no. 4, pp. 451–462, Nov. 2000.
[3] K. Tin-Yau and Y. Dit-Yan, "Constructive algorithms for structure learning in feedforward neural networks for regression problems," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 630–645, May 1997.
[4] R. Parekh, J. Yang, and V. Honavar, "Constructive neural-network learning algorithms for pattern classification," *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 436–451, Mar. 2000.
[5] M. Islam, A. Sattar, F. Amin, Y. Xin, and K. Murase, "A new adaptive merging and growing algorithm for designing artificial neural networks," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 39, no. 3, pp. 705–722, Jun. 2009.
[6] M. M. Islam, M. A. Sattar, M. F. Amin, Y. Xin, and K. Murase, "A new constructive algorithm for architectural and functional adaptation of artificial neural networks," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 39, no. 6, pp. 1590–1605, Dec. 2009.
[7] R. Reed, "Pruning algorithms-a survey," *IEEE Trans. Neural Netw.*, vol. 4, no. 5, pp. 740–747, Sep. 1993.

[8] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990, pp. 524–532.
[9] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 54–65, Jan. 1994.
[10] F. Jian and X. Yugeng, "Neural network design based on evolutionary programming," *Artif. Intell. Eng.*, vol. 11, no. 2, pp. 155–161, Apr. 1997.
[11] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 694–713, May 1997.
[12] P. P. Palmes, T. Hayasaka, and S. Usui, "Mutation-based genetic neural network," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 587–600, May 2005.
[13] J. H. Ang, K. C. Tan, and A. Al-Mamun, "Training neural networks for classification using growth probability-based evolution," *Neurocomputing*, vol. 71, nos. 16–18, pp. 3493–3508, Oct. 2008.
[14] D. Whitley, T. Starkweather, and C. Bogart, "Genetic algorithms and neural networks: Optimizing connections and connectivity," *Parallel Comput.*, vol. 14, no. 3, pp. 347–361, Aug. 1990.
[15] D. Whitley and T. Starkweather, "Optimizing small neural networks using a distributed genetic algorithm," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 1. 1990, pp. 206–209.
[16] A. C. Martinez-Estudillo, C. Hervas-Martinez, F. J. Martinez-Estudillo, and N. Garcia-Pedrajas, "Hybridization of evolutionary algorithms and local search by means of a clustering method," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 36, no. 3, pp. 534–545, Jun. 2005.
[17] F. H. F. Leung, H. K. Lam, S. H. Ling, and P. K. S. Tam, "Tuning of the structure and parameters of a neural network using an improved genetic algorithm," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 79–88, Jan. 2003.
[18] P. A. Gutiérrez, C. Hervás-Martínez, and F. J. Martínez-Estudillo, "Logistic regression by means of evolutionary radial basis function neural networks," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 246–263, Feb. 2011.
[19] X. Yao, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, no. 9, pp. 1423–1447, Sep. 1999.
[20] J. Yaochu and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Trans. Syst., Man, Cybern., Part C: Appl. Rev.*, vol. 38, no. 3, pp. 397–415, May 2008.
[21] J. C. F. Caballero, F. J. Martinez, C. Hervas, and P. A. Gutierrez, "Sensitivity versus accuracy in multiclass problems using memetic Pareto evolutionary neural networks," *IEEE Trans. Neural Netw.*, vol. 21, no. 5, pp. 750–770, May 2010.
[22] A. Kaylani, M. Georgiopoulos, M. Mollaghasemi, G. C. Anagnostopoulos, C. Sentelle, and Z. Mingyu, "An adaptive multiobjective approach to evolving ART architectures," *IEEE Trans. Neural Netw.*, vol. 21, no. 4, pp. 529–550, Apr. 2010.
[23] W. Spears and V. Anand, "A study of crossover operators in genetic programming," in *Methodologies for Intelligent Systems*, vol. 542, Z. Ras and M. Zemankova, Eds. Berlin, Germany: Springer-Verlag, 1991, pp. 409–418.
[24] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. New York: Wiley, 1966.
[25] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Piscataway, NJ: IEEE Press, 1995.
[26] B. M. Wilamowski, "Neural network architectures and learning algorithms," *IEEE Ind. Electron. Mag.*, vol. 3, no. 4, pp. 56–63, Dec. 2009.
[27] B. M. Wilamowski, N. J. Cotton, O. Kaynak, and G. Dundar, "Computing gradient vector and Jacobian matrix in arbitrarily connected neural networks," *IEEE Trans. Ind. Electron.*, vol. 55, no. 10, pp. 3784–3790, Oct. 2008.
[28] L. Prechelt, "Proben1 – A set of neural network benchmark problems and benchmarking rules," Fakultat Inf., Univ. Karlsruhe, Karlsruhe, Germany, Tech. Rep. 21/94, Sep. 1994.
[29] D. B. Fogel, "An introduction to simulated evolutionary optimization," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 3–14, Jan. 1994.
[30] X. Yao, "An empirical study of genetic operators in genetic algorithms," *Microprocess. Microprogram.*, vol. 38, nos. 1–5, pp. 707–714, Sep. 1993.
[31] J. Reed, R. Toombs, and N. Barricelli, "Simulation of biological evolution and machine learning," *J. Theoret. Biol.*, vol. 17, no. 1, pp. 319–342, 1967.
[32] Y. Xin, L. Yong, and L. Guangming, "Evolutionary programming made faster," *IEEE Trans. Evolut. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.
[33] *UCI Machine Learning Repository* [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html
[34] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006.
[35] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Proc. 2nd Int. Symp. Inf. Theory*, vol. 1. 1973, pp. 267–281.

Tatt Hee Oong received the B.Eng. degree in electronic engineering (with first class honors) from Universiti Sains Malaysia (USM), Minden, Malaysia, in 2010. He is currently pursuing the Ph.D. degree in electrical and electronic engineering and is part of the Imaging and Intelligent Systems Research Team, School of Electrical and Electronic Engineering, USM.

His current research interests include neural networks, pattern recognition, and evolutionary algorithms.

Nor Ashidi Mat Isa (M'10) received the B.Eng. degree in electrical and electronic engineering (with first class honors) and the Ph.D. degree in electronic engineering (majoring in artificial neural networks and image processing) from Universiti Sains Malaysia (USM), Minden, Malaysia, in 1999 and 2003, respectively.

He is currently a Lecturer with the School of Electrical and Electronics Engineering, USM. He and his research team (Imaging and Intelligent System Research Team) have published their works nationally and internationally. Their contributions can be found in numerous international and national journals, chapters in books, and in international and national proceedings. His current research interests include intelligent systems, image processing, neural networks, biomedical engineering (i.e., intelligent diagnostic systems), and algorithms.