An Adaptive Multiobjective Approach to Evolving ART Architectures

Assem Kaylani, Michael Georgiopoulos, *Senior Member, IEEE*, Mansooreh Mollaghasemi, Georgios C. Anagnostopoulos, *Member, IEEE*, Christopher Sentelle, and Mingyu Zhong

***Abstract*—In this paper, we present the evolution of adaptive resonance theory (ART) neural network architectures (classifiers) using a multiobjective optimization approach. In particular, we propose the use of a multiobjective evolutionary approach to simultaneously evolve the weights and the topology of three well-known ART architectures; fuzzy ARTMAP (FAM), ellipsoidal ARTMAP (EAM), and Gaussian ARTMAP (GAM). We refer to the resulting architectures as MO-GFAM, MO-GEAM, and MO-GGAM, and collectively as MO-GART. The major advantage of MO-GART is that it produces a number of solutions for the classification problem at hand that have different levels of merit [accuracy on unseen data (generalization) and size (number of categories created)]. MO-GART is shown to be more elegant (does not require user intervention to define the network parameters), more effective (of better accuracy and smaller size), and more efficient (faster to produce the solution networks) than other ART neural network architectures that have appeared in the literature. Furthermore, MO-GART is shown to be competitive with other popular classifiers, such as classification and regression tree (CART) and support vector machines (SVMs).**

***Index Terms*—ARTMAP, category proliferation, classification, genetic algorithms (GAs), genetic operators, machine learning.**

I. Introduction

ADAPTIVE RESONANCE THEORY (ART) was developed by Grossberg [1]. Some of the ART architectures that have appeared in the literature include fuzzy ARTMAP (FAM) [2], ellipsoidal ARTMAP (EAM) [3], and Gaussian ARTMAP (GAM) [4]. All of these ART architectures possess a number of desirable properties, such as they can solve arbitrarily complex classification problems, they converge quickly to a solution (within a few presentations of the list of input/output patterns belonging to the training set), they have the ability to recognize novelty in the input patterns presented to them, they can operate in an online fashion (new input patterns can be learned by the ART system without retraining with the old input/output patterns), and they produce answers that can be explained with relative ease.

One of the limitations of these ART architectures, which has been repeatedly reported in the literature, is the category proliferation problem, where a relatively large number of categories are needed to represent the training data. Another limitation is their performance dependence on the parameters chosen for training the data. The choice of these parameters requires good knowledge of these architectures and often requires some experimentation to get the best results using these architectures, a computationally intensive proposition.

To alleviate these problems, we introduced genetic fuzzy ARTMAP (GFAM) in [5] and [6]. GFAM uses a genetic algorithm (GA) (see [7]) to evolve simultaneously the weights, as well as the topology of ART neural networks. In [6], we extended the ideas of genetically engineering FAM (GFAM, in [5]) to EAM and GAM, and introduced several improvements, which resulted in significant gains in terms of efficiency.

Genetic ART (GART) starts with a population of trained ART networks, with the number of hidden-layer nodes and the interconnection weights to these nodes fully determined (at the beginning of the evolution) by ART's training rules. To this initial population of ART networks, GA operators are applied to modify these trained ART architectures (i.e., number of nodes in the hidden layer, and values of the interconnection weights) in a way that encourages better generalization and smaller size architectures. The optimization problem in GART has two objectives: maximize classification accuracy on a validation set, and minimize network complexity (size of the network), measured in terms of the number of hidden nodes (categories). In GART, these two objectives were combined using a weighted sum fitness function. A problem with this approach is that the user has to *a priori* specify their preference of accuracy and complexity, by choosing the weights in this fitness function. However, choosing good weights for the fitness function is a data-dependent problem. To overcome this, the user should run the algorithm for different settings of the weights in the fitness function, an expensive proposition. Furthermore, the basic weighted sum approach, used in GART, might *not* be able to reproduce all possible solutions that might be of interest to the user. Although this is generally true for a weighted sum approach that uses constant weights, in defining a single-objective function from multiple objectives, Jin *et al.* [8] have eloquently demonstrated that a dynamic weight aggregation approach overcomes this limitation and have used this approach in a multiobjective training of multilayer perceptrons [9], maximizing accuracy and minimizing complexity. However, another way of alleviating

Manuscript received August 16, 2008; revised November 01, 2009; accepted November 06, 2009. Date of publication February 17, 2010; date of current version April 02, 2010. This work was supported in part by the National Science Foundation (NSF) under Grants 0341601, 0647018, 0717674, 0717680, 0647120, 0525429, 0806931, and 0837332.

A. Kaylani, M. Georgiopoulos, C. Sentelle, and M. Zhong are with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32826 USA (e-mail: akaylani@gmail.com; michaelg@mail.ucf.edu; csentelle@cfl.rr.com; myzhong@ucf.edu).

M. Mollaghasemi is with the Industrial Engineering, University of Central Florida, Orlando, FL 32826 USA (e-mail: mollagha@mail.ucf.edu).

G. C. Anagnostopoulos is with the Electrical and Computer Engineering, Florida Institute of Technology, Melbourne, FL 32901-6975 USA (e-mail: georgio@fit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2009.2037813

GART's limitation, and the focus in this paper, is to use an appropriate multiobjective evolutionary algorithm (MOEA) to find multiple solutions to the multiobjective problem at hand.

In particular, in this paper, we propose an improved GA for the evolution of ART architectures. The proposed GA relies on adaptive parameter control. This has the advantage of avoiding the reliance of the performance on genetic parameters that were introduced in GFAM such as the probability of mutation and probability of deletion of a FAM category. Therefore, the proposed approach avoids the need for experimentation to achieve the best possible performing ART network. Furthermore, it alleviates the need to tweak the GA algorithm parameters in order to achieve good results. The GA algorithm parameters are automatically adapted to fit the classification problem at hand, resulting in a classifier that does not require any user intervention. Furthermore, the proposed improved GA approach relies on multiobjective optimization techniques (thus overcoming the issues associated with framing a multiobjective optimization problem as a single-objective optimization problem, mentioned above), while it chooses the best GA parameters in an adaptive, problem-dependent, fashion. We named this improved GA approach as MO-GART, and more selectively, MO-GFAM, MO-GEAM, and MO-GGAM.

Using genetic algorithms to optimize performance has not been limited to classifiers, such as ART. Ishibuchi *et al.* have produced a good number of papers, dated back in 1994 [10], where genetic algorithms were used to design fuzzy classifiers. In particular, in 1995, Ischibuchi *et al.* [11] used a genetic approach with a single-objective fitness function to design a fuzzy classifier that maximized the classification accuracy (one of the objectives) and minimized the number of square fuzzy rules (the other objective); a very similar paper [10] designed a fuzzy classifier with rectangular fuzzy rules instead of square fuzzy rules, resulting in a fuzzy classifier with a reduced fuzzy rule size. In 1997, Ishibuchi *et al.* [12] compared a number of genetic-based single-objective approaches and a multiobjective approach to design a fuzzy classifier that maximizes classification accuracy and minimizes the number of produced linguistic rules. The multiobjective method selects half of the population from one generation to the next using a single-objective function with random weights (not constant weights); this way it drives the population in many different directions in the objective space. In one of the variations of the multiobjective approach, the confidence of a fuzzy rule is taken into consideration in making classification decisions. It turned out that the multiobjective approach that incorporates the confidence of the fuzzy rule produced the best results. In [13], Ishibuchi *et al.* consider a three-objective optimization problem where the objective is to maximize classification, minimize the number of fuzzy rules, and minimize the number of antecedent rules; in their effort, they consider two approaches of generating the fuzzy rules (a rule-based approach and a GA-based approach) and they report good results on the Wine classification problem, a problem that has a large number of attributes (13 attributes). In a companion paper [14] Ishibuchi *et al.* consider, once more, the three-objective optimization problem of maximizing the correctly classified training patterns, minimizing the number of fuzzy rules, and minimizing the total number of antecedent conditions. In their work, they consider a rule selection approach where a small number of rules is extracted from a larger number of candidate rules, and a fuzzy-genetic-based machine learning approach where the candidate rules are produced by the genetic algorithm. Through a number of simulations, the authors demonstrated that a small number of interpretable rules can be produced, through both of these approaches, on a number of select classification problems. In one of their most recent publications [15], Ishibuchi *et al.* examine the interpretability-accuracy tradeoff in fuzzy-rule-based classifiers using a multiobjective fuzzy-genetics-based machine learning (GBML) algorithm. This analysis revealed that each data set has a different relation between complexity of fuzzy-rule-based classifiers and their generalization ability for test patterns. Another interesting result from this paper is that in some cases lower error rates were obtained by the single-objective formulation than by the multiobjective formulation, suggesting the need to improve the search ability of evolutionary multiobjective optimization (EMO) algorithms.

This short review indicates that there is a very rich literature on multiobjective optimization of fuzzy classifiers, pioneered by Ishibuchi *et al.* , but a literature that has many more contributors as the evolutionary multiobjective optimization of fuzzy-rule-based systems bibliography page by Cococcioni demonstrates.[1] The literature is also rich in contributions related to multiobjective optimization of neural networks. For instance, Everson and Fieldsend [16] have provided a multiclass receiver operating characteristic (ROC) analysis for a three-class problem using a k-nearest neighbor and a multilayer classifier. Jin *et al.* [17] optimize classification accuracy and connectivity within the framework of spiking neural networks. Graning *et al.* [18] show on a few benchmark problems that multiobjective optimization of the structure and the weights of multilayer neural networks, solving binary classification problems, results in networks that better generalize compared to the case where the objectives are optimized through a single-objective optimization approach. Jin and Sendhoff provide a very comprehensive review of Pareto-based multiobjective machine learning approaches in [19]. There they illustrate that three benchmark problems (generating interpretable models, model selection for generalization, and ensemble generation) can benefit from a Pareto-based multiobjective approach.

The aforementioned references provide an incomplete review of the work that has been conducted in the literature to design classifiers or fuzzy classifiers, using an evolutionary multiobjective optimization approach. Despite that fact that the review is not complete, our specific focus on the multiobjective optimization of ART classifiers revealed no prior work in the literature beyond the research mentioned in the beginning paragraphs of the Introduction, all of which used a single-objective function approach to optimize the two objectives of maximizing classification accuracy and reducing the classifier's size. This paper's intent is to design an ART classifier using a genetic-based multiobjective optimization approach. Multiobjective optimization

[1]http://www2.ing.unipi.it/~g000502/emofrbss.html

approaches have been generally reported as being superior to single-objective optimization approaches in solving problems with multiple objectives. Our approach is a Pareto-based approach, and as Jin and Sendhoff emphasize [19], Pareto-based approaches offer a deeper insight into the learning problem by analyzing the Pareto front. Furthermore, our approach frees the user from the difficult task of *a priori* choosing the ART network parameters to optimize ART's performance (classification accuracy and size); on the contrary, it requires no *a priori* setting of ART network parameters by the user, quite often a time consuming task.

In our work, the MO-GART family of proposed classifiers is compared to other ART classifiers, and non-ART classifiers. In particular, we compare MO-GART with GART, ssFAM, ssEAM, ssGAM (see [20]), and a few other non-ART-based classifiers. This comparison is based on the accuracy, size, and computational complexity of the classifiers. Our results show that MO-GFAM, MO-GEAM, and MO-GGAM perform well on a number of classification problems, and optimally on some of these problems. For instance, MO-GART attained a better generalization performance and required a smaller than, or equal, sized network (in almost all problems tested), compared to ssFAM, ssEAM, and ssGAM networks, while also requiring reduced (sometimes significantly reduced) computational effort to achieve these advantages. Furthermore, MO-GART attained a better generalization performance (in a number of problems), and was smaller than or equal in size compared to GART (in most problems), while needing reduced computational efforts to achieve these advantages (in all problems). Also, we found that MO-GART was less computationally demanding, and of smaller size, while achieving an accuracy as competitive as its counterpart support vector machine (SVM) classifier. Finally, we discovered that MO-GART produces more accurate classifiers, of competitive sizes, than classification and regression tree (CART) [21], at the expected expense of more computational effort. It is worth noting that neither GART nor MO-GART require *a priori* setting of the ART network parameters, an important advantage from the user's perspective. It is also worth noting that the only GA parameters that need to be defined by GART or MO-GART user are the population size, a mutation constant, and an upper limit on the number of evolutions all of which are set to default values that work well for all the data sets that we have experimented with. Finally, as mentioned earlier, to the best of our knowledge, this is the first attempt to design ART networks, using a multiobjective optimization approach.

The organization of this paper is as follows. In Section II, we present a rather lengthy, but needed (in our opinion), exposition of the basics of ART architectures (only the elements needed for this paper are presented in Section II). Furthermore, in Section II, we provide the associated literature related to adaptation of the GA parameters and multiobjective optimization techniques that motivate the choices made in our proposed GA method, delineated in Section III. In Section III, we describe all the important elements pertinent to the multiobjective evolution of the ARTMAP architectures that correlate with the choices suggested by the literature and our prior experiences with GART [6]. In Section IV, we discuss the experiments conducted, the data sets used in the experimentation, and we provide the performance comparisons between the MO-GART and other ART-based classifiers (ssART, GART), as well as non-ART-based classifiers (SVM and CART). In Section V, we analyze, in extensive detail, some of the pairwise comparisons of the MO-GART results with ssART, GART, SVM, and CART. Furthermore, the sensitivity of the MO-GART results to the seed parameter for all the data sets that we experimented with, as well as visualization of the Pareto fronts of MO-GART, GART, and ssART on select data sets are also contained in Section V. In Section VI, we summarize our contributions and findings.

II. Preliminaries

A. The ARTMAP Architectures

Grossberg [1] introduced the foundation of ART in 1976. Later, based on that work, ART1 was developed to perform clustering (self-organizing) of binary patterns [22]. ART1 was then extended to ART2 to handle real-valued patterns [23]. In 1991, Carpenter and Grossberg introduced ARTMAP [24], which was capable of performing classification of binary patterns. They then simplified the ART2 architecture and introduced an improved version called fuzzy ART [25]. Furthermore, in 1992, Carpenter and Grossberg extended ARTMAP to fuzzy ARTMAP, which is capable of classifying real-valued input patterns [2].

Since the introduction of fuzzy ARTMAP, other ART architectures have been introduced into the literature. The focus of this effort is on fuzzy ARTMAP and two other ART architectures: ellipsoidal ARTMAP (see [3]) and Gaussian ARTMAP (see [4]). The objective in this effort is to illustrate how genetically engineered ART architectures can be designed from a population of fuzzy ARTMAPs, ellipsoidal ARTMAPs, or Gaussian ARTMAPs. It is assumed that the reader is familiar with these ART architectures. This section only describes the specifics of ART architectures needed to understand the genetically engineered ART structures. For simplicity, these ART architectures are referred to as ART and their specific name is used (FAM or EAM or GAM) only when there is a need to differentiate one from the other.

1) Operation of ART: An ART architecture consists of an input layer, where the inputs are applied, a category representation layer, where categories (compressed representations of the input patterns are formed), and an output layer where the correct mapping between the input patterns and their associated labels are established. Training in ART is achieved by presenting a training set to the ART network. Given a set of inputs and associated label pairs $\mathbf{I}_1, \text{label}(\mathbf{I}_1), \mathbf{I}_2, \text{label}(\mathbf{I}_2), \ldots, \mathbf{I}_{PT}, \text{label}(\mathbf{I}_{PT})$ (called the training set), we want to train ART to map every input pattern of the training set to its corresponding label. To achieve the aforementioned goal, we present the training set to the ART architecture repeatedly, as many times as it is necessary for ART to correctly classify these input patterns. The task is considered accomplished (i.e., learning is complete) when the weights in ART do not change during a training set presentation, or after a specific number of list presentations is reached.

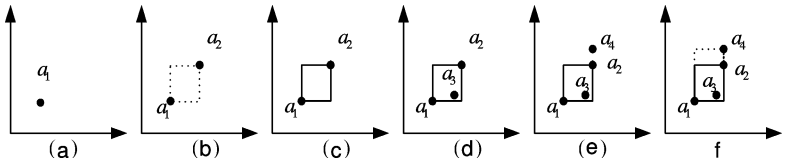

Fig. 1. FAM learning (2-D example). (a) Category with 0 size. (b) Introducing a new pattern $\mathbf{I}_2$, represented by $\mathbf{a}_2$. (c) Category expands to include $\mathbf{a}_2$. (d) Since new pattern $\mathbf{I}_3$, represented by $\mathbf{a}_3$ is inside the category, it does not change its size. (e) New pattern $\mathbf{I}_4$, represented by $\mathbf{a}_4$ is presented. (f) Since $\mathbf{a}_4$ is outside the category, the category is expanded to include $\mathbf{a}_4$, within its boundaries.

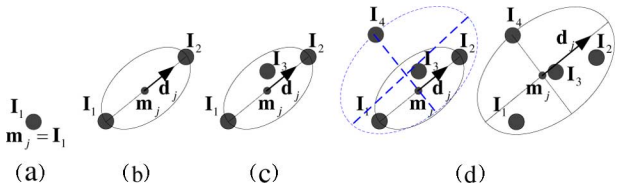

Fig. 2. EAM learning (2-D example). (a) Category with 0 size. (b) Introducing a new pattern $\mathbf{I}_2$; the category expands to include $\mathbf{I}_2$. (c) Introducing a new pattern $\mathbf{I}_3$; since the category includes $\mathbf{I}_3$, it does not change its size. (d) Pattern $\mathbf{I}_4$ is presented; since this pattern is outside the category, the category is expanded to include $\mathbf{I}_4$ within its boundaries.

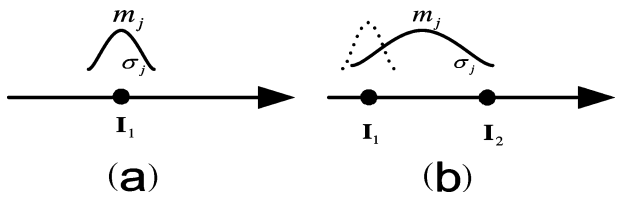

Fig. 3. GAM learning (1-D example). (a) Category with 0 size. (b) Introducing a new pattern $\mathbf{I}_2$; the category characteristics (mean, standard deviation, of the Gaussian curve, as well as number of points encoded by the Gaussian curve) change to include the new knowledge that the new input pattern brings.

The weights in ART correspond to compressed representations of the input patterns presented to the ART network during its training phase. These compressed representations have a geometrical interpretation. In particular, every node (category) in the category representation layer of FAM has weights that completely define the lower and upper endpoints of a hyperbox. At the beginning of training, every category of FAM starts as a trivial hyperbox (equal to a point), and subsequently, it expands to incorporate within its boundaries all the input patterns that in the training phase choose this hyperbox as their representative hyperbox, and are encoded by it (see Fig. 1, where the category expansion of FAM is shown for an example data set). The size of hyperbox is measured as the sum of the lengths of its sides.

Also, every node (category) in the category representation layer of EAM has template weights that completely define an ellipsoid, through its center, direction of major axis, length of the major axis, and ratio of lengths of minor axes to major axis in the ellipsoid. At the beginning of training, every EAM category starts as a trivial ellipsoid (equal to a point), and subsequently, it expands to incorporate within its boundaries all the input patterns that in the training phase chose this ellipsoid as their representative ellipsoid, and are encoded by it (see Fig. 2, where the category expansion of EAM is shown for an example data set). The size of the ellipsoid is measured as the length of the major axis.

Finally, every node (category) in the category representation layer of GAM has template weights that define the mean vector, the standard deviation vector of a multidimensional Gaussian distribution, and the number of points that are associated with this Gaussian distribution. At the beginning of training, every category of GAM starts as a collection of Gaussian distributions in every dimension, with mean equal to the input pattern that was first encoded by this category, and a small standard deviation vector [Fig. 3(a)]; as training progresses in GAM, this GAM category is modified to incorporate the information of the additional input patterns that are encoded by it [see Fig. 3(b) for an illustration of how the GAM category is modified for an example data set]. At any point in time, the mean vector of this Gaussian distribution, corresponding to a category, is equal to the mean vector of all the input patterns encoded by the category, and the variance vector of the Gaussian distribution is equal to the variance vector corresponding to the input patterns that were encoded by the category, while the number of the points associated with this Gaussian distribution is the number of points that chose this category as their representative category.

It is also worth mentioning that the categories in FAM, EAM and GAM are allowed to expand up to a point allowed by a threshold, controlled by a network parameter denoted as the baseline vigilance parameter $(\bar{\rho}_a)$. This parameter assumes values in the interval $[0, 1]$. Small values of this parameter allow the creation of large categories, while large values of this parameter allow the creation of small categories. In the one extreme when $\bar{\rho}_a$ is equal to 0, a FAM or EAM category, equal to the whole input space, could be created, while at the other extreme when $\bar{\rho}_a$ is equal to 1 only point categories are formed. In GAM, small values of this parameter allow more and more patterns to be encoded by a GAM category, while large values of this parameter allow only a few patterns to be encoded by a GAM category. It turns out that this parameter has a significant effect on the number and type of categories formed, and consequently, it affects the performance of these networks.

The performance of ART networks is measured in terms of the number of categories created in its training phase (small number of categories is good), and how well it generalizes on unseen data (high generalization accuracy is good). The performance of an ART architecture depends on the choice of the vigilance parameter. It has been a known fact that performance in ART is also affected by the order according to which training data are presented to an ART architecture.

2) ART Category Proliferation Problem: One of the limitations of these ART architectures that has been repeatedly reported in the literature is the category proliferation problem. This refers to the creation of a relatively large number of categories to represent the training data. The creation of a large number of categories means poor compression of the training data. Quite often, the category proliferation problem, observed

in ART, is connected with the issue of overtraining. Overtraining occurs when ART is trying to learn the training data perfectly at the expense of degraded generalization performance (i.e., classification accuracy on unseen data) and also at the expense of creating many categories to represent the training data (leading to the category proliferation problem). Also, it has been related to several limitations of ART, such as the representative inefficiency of the categories or the excessive triggering of the match tracking mechanism due to existence of noise.

Since the early 1990s, a number of ART modifications and improvements have been published in the literature. Authors tried to improve the ART learning properties, speed up ART's training, and address the ART category proliferation problem. A variety of innovations have been introduced to combat the category proliferation problem in ART. For instance, PROBART [26] eliminates the match tracking mechanism and instead stores probability information in the map field. In 1996, GAM was introduced [4]. The authors attribute the category proliferation to two causes: sensitivity to noise and inefficiency of FAM categories. GAM operates in a similar fashion as FAM but it relies on a Gaussian-based measure of similarity in its operation, therefore eliminating the reliance on the inefficient category shapes of FAM and reducing the sensitivity to noise. Micro-ARTMAP [27] suppresses the match tracking mechanism and employs a probabilistic map to encourage creation of large categories, and hence reduce their number. Safe micro-ARTMAP [28], introduced later, adds a mechanism to limit the growth of a category in response to a single pattern. Semisupervised learning of ART was introduced [20] in 2003, where it allows, with a certain tolerance, categories to encode patterns that are not mapped to the same label. This reduces the sensitivity to noise and hence the number of categories created. Three semisupervised architectures were introduced in [20]: ssFAM, ssGAM, and ssEAM. In [29], Koufakou *et al.* suggest the use of cross validation to prevent overtraining, and therefore, avoid the creation of unnecessary categories. In addition, the work by Carpenter *et al.*[30], Williamson [31], and Parrado-Hernadez *et al.* [32] proposes that the ART structure is changed from a winner-take-all to a distributed version and slow learning is employed with the intent of creating fewer ART categories and reducing the effects of noisy patterns. To address the inefficiency of FAM categories in data representation, EAM was introduced in 2001 [3]. EAM was similar to FAM except that it relied on category regions defined as hyper-ellipsoids rather than FAM's original hyper-rectangles.

B. Adaptation in Genetic Algorithms

When applying a GA to solve an optimization problem, we not only need to choose the algorithm, representation, and operators for the problem, but we also need to choose parameter values and operator probabilities for the GA so that it will not only find the solution, but also find the solution efficiently. In many cases, researchers choose these parameter values and operator probabilities based on experience or experimentation on a specific problem. The ways to choose the GA parameters and operator probabilities has attracted a lot of interest. A number of researchers suggested good parameter values as a result of extensive experimentation on a range of optimization problems. For example, Jong [33] proposed to set $p_m = 0.001$ and $p_c = 0.6$, where p_m and p_c stand for the probability of mutation and crossover, respectively. In [34], Muhlenbein and Schlierkamp-Voosen proposed $p_m = 1/\beta$, and in [35], Back proposed $p_m = 1.75/n * \beta^{1/2}$, where n represents the population size and β represents the bit length of a chromosome.

Finding good GA parameter values for a certain optimization problem is a time consuming process. It is prone to human error, which may lead to suboptimal results. Moreover, what might be initially considered as a good parameter setting could, in the progress of evolution, prove not to be as good, since the search may move to different regions of the solution space. As a result, an emphasis was placed on designing GAs where the parameters automatically adapt to the problem at hand.

The GA parameters that affect its performance include environmental parameters such as population size and the objective (fitness) function. The adaptation can be applied to global parameters such as mutation rate, mutation strength, or crossover rate that are affecting all individuals in the population, or applied to local parameters where the parameter value is customized for each individual. Also, some of the research proposes the customization of the parameter setting at the component level (part of the individual).

The majority of research, though, focuses on adapting the mutation rate or mutation strength. For real-valued representations, the term mutation strength, sometimes called mutation step size, refers to the magnitude of change in each mutated variable. This is different in binary representation schemes, where the term mutation rate is used to express how probable it is for a certain binary variable to be changed (inverted).

The adaptation approaches can be distinguished in three groups, in order of increasing complexity: *deterministic*, *adaptive*, and *self-adaptive* [36]. Each of these approaches is described below.

- *Deterministic:* Deterministic adaptation refers to the dynamic adjustment of a GA parameter using a deterministic rule, and without feedback from the quality of the solution achieved by the evolutionary process. This rule can be based on a schedule (similar to simulated annealing) or number of generations (see [37]). In general, the objective in this approach is to alter the GA parameters in such a way that result in a widespread search at the beginning of the optimization, and increasingly localized search at later stages. An example of this approach can be found in [38], where the mutation step sizes are discounted by a constant factor each time an offspring is produced.
- *Adaptive:* This approach uses some form of feedback from the GA that is used to determine the direction and/or magnitude of the change to the GA parameter. In [39], the standard deviation of the Gaussian mutation was varied based on a temperature parameter. The temperature parameter was defined as $T(\eta) = 1 - (f(\eta)/f_{\max})$, where $f_{\max}$ is the maximum fitness for a given task. Thus, the temperature of a solution is determined by how close the solution is to being an optimal solution for the task under consideration. Solutions with a high temperature are mutated severely, and those with a low temperature are mutated only slightly. This allows a coarse-grained search initially, and

a progressively finer grained search as the GA approaches a solution for the assigned task.

In [40], the authors suggest the use of a feedback signal that is defined by the difference $\mathrm{fit}_{\max}(P,t) - \mu(P,t)$, where $\mathrm{fit}_{\max}(P,t)$ is the maximum fitness and $\mu(P,t)$ is the average fitness of solutions in population $P(t)$ at generation t. This difference is used as an indication of closeness to convergence. This difference is likely to be less for a population that has converged to an optimal solution than that for a random population, scattered in the solution space. The authors define the adaptive rates of crossover and mutation for all chromosomes to be inversely proportional to this difference. To make this mechanism less disruptive for good solutions, the mechanism is adjusted to have low values of parameters for high-fitness values and high values of parameters for low-fitness values, as follows: $p_m(x) = k_1(\mathrm{fit}_{\max}(P,t) - \mathrm{fit}(x,t)/\mathrm{fit}_{\max}(P,t) - \mu(P,t))$, and $p_c = k_2(\mathrm{fit}_{\max}(P,t) - \mathrm{fit}(x,t)/\mathrm{fit}_{\max}(P,t) - \mu(P,t))$. Therefore, in this case, the parameters are controlled at the individual level.

- *Self-Adaptive:* In this approach, the GA parameters undergo evolution. The GA parameters are encoded in the chromosomes and evolved as part of the solution. The encoded parameters will lead to better fitness for individuals with better parameter values; and since these individuals are more likely to survive and reproduce, this mechanism will propagate these better parameter values.

In our implementation of genetic ARTMAP, we chose to use an adaptive approach, where a feedback signal is used to determine the operator probability at the component level.

C. Multiobjective Evolutionary Algorithms

Many real-world problems involve simultaneous optimization of conflicting objectives. This is the basic challenge of multiobjective optimization research. Evolutionary algorithms have been used extensively to solve multiobjective optimization problems, resulting in a body of knowledge known as MOEAs. A number of authors have published surveys of MOEA, such as [41], and the reader can find more details about MOEAs there.

With conflicting multiple objectives, there is no single optimal solution, but rather, there are a set of good solutions. It is often desirable to find these good solutions as they provide alternative solutions to the problem at hand. Evolutionary algorithms (EAs) are suitable for solving multiobjective optimization problems because EAs are population-based search algorithms, and, as such, they can find, in a single run, multiple good solutions on the surface defined by the multiple objectives that are to be optimized.

A Pareto-optimal solution is a solution that is not dominated by any other solution in the search space. The entire set of such optimal solutions is often referred to as the Pareto front. The main focus of most MOEA research is to minimize the distance of the generated solutions to the true Pareto front and to maximize the coverage (diversity) of the discovered Pareto set. A good Pareto set may be obtained by appropriate guiding of the search process through careful design of the selection and fitness assignment strategies, which is the main challenge concerning multiobjective optimization using GAs. The selection operator (in single-objective and multiobjective optimization problems) determines the solutions that will be selected for the reproduction of the next generation. The selection operation emphasizes fit individuals in the population by giving them a higher chance to breed. The selection scheme should emphasize the characteristics of good solutions in order for the evolutionary process to produce better solutions in successive generations. In the presence of multiple objectives, the determination of "better solutions" is not as straightforward as it is in the single-objective case. In review, the objective of the selection operation in a multiobjective problem is to lead the evolutionary process to a set of optimal solutions, rather than one optimal solution as in the case of single-objective problems.

One of the simplest approaches for dealing with a multiobjective problem is to convert the problem into a single-objective problem. This is done by implementing a mechanism that combines the multiple objectives into a single objective. These approaches try to converge to a specific point on the Pareto front. Therefore, the combining mechanism determines the relative importance of the objectives. In these methods, to generate the entire Pareto front, the analyst must perform multiple runs and vary the conditions of the combining mechanism. The simplest method for combining the objective is the weighted sum approach. This can be expressed as $\mathrm{fit}(x) = \sum_{i=1}^{L} w_i f_i(x)$ where L denotes the number of objectives and $f_i(x)$ denotes the ith objective function. This is the approach that was adopted in GART [42] and a number of other evolutionary neural networks such as [43]. It follows immediately that the solution that optimizes $\mathrm{fit}(x)$ is a Pareto optimal point, since if not, then there must exist a feasible x which improves on at least one of the objectives without compromising the others and hence produces a smaller value of the weighted sum. It is necessary to scale or normalize the objectives to avoid having one objective dominate the others. This requires knowledge of the range of each objective, which might not be available for many real-world applications.

The weighted sum of the objectives fitness function approach has a number of drawbacks. The first drawback is that this scheme is not able to generate the nonconvex regions of the Pareto front for any combination of the weights. This has been pointed out by a number of researchers, such as [44]. The second drawback is that the solutions, selected by evenly varying the weights, are not guaranteed to be evenly distributed on the Pareto front. This results in a poor diversity of the Pareto solutions found. As these drawbacks were identified in GART [6], in this paper, we look for a multiobjective approaches that overcome these drawbacks. Some of the drawbacks of the single objective with constant weights function have been addressed by the dynamic weight aggregation approach introduced in [8], however the focus in this paper is to address the aforementioned issues by emphasizing a multiobjective optimization to tackle them.

The more recent research in multiobjective optimization avoids combining the objectives into a single objective. Rather, they treat objectives separately, and solutions are evaluated with respect to each one of the objectives at every generation. Therefore, these approaches are more suited for finding multiple Pareto solutions. These approaches do not normally require a mechanism that determines the relative importance of

objectives. The aggregation methods of selection, mentioned above, are often referred to as *a priori* methods because they normally incorporate preference beforehand. Alternatively, the methods that attempt to produce the whole Pareto front and give the option to the user to decide from a set of optimal solutions are referred to as *a posteriori* methods.

One early example of the *a posteriori* method is the pioneering work of Schaffer [45] where vector evaluated genetic algorithm (VEGA) was introduced. In VEGA, the selection step generates a number of subpopulations by performing proportional selection according to each objective in turn. Then, these subpopulations are combined to obtain a new population, on which the genetic operators, crossover and mutation, are applied. VEGA has a major drawback in that its selection scheme is biased towards some Pareto optimal solutions. To overcome problems identified in VEGA, Goldberg [7] suggested ranking of solutions based on their Pareto optimality. In this scheme, Pareto optimal solutions are equally assigned the highest fitness, and therefore, they have increased chance of survival and breeding. The rest of the population is assigned fitness values that depend on their closeness to the Pareto front. A number of authors proposed algorithms based on this ranking scheme, such as Srinivas [46] who introduced the nondominated sorting genetic algorithm (NSGA) and Fonseca [47] who introduced the multiobjective genetic algorithm (MOGA). The niched-Pareto genetic algorithm (NPGA; see [48]) uses a tournament selection scheme based on Pareto dominance. In a similar fashion, as a successor to NPGA, a revised version was introduced and referred to as NPGA2 (see [49]).

Elitism is a selection mechanism that aims at preserving good performance through successive generations. In multiobjective optimization, elitism refers to preserving nondominated solutions found along the evolutionary process. Elitism has been adopted in the more recent MOEA research. For example, Ishibuchi [50] uses a random weighted sum approach, with elitism, to produce the Pareto front. The weights are generated randomly each time an individual is selected. The nondominated set of solutions is stored externally and updated every generation. The nondominated sorting genetic algorithm II (NSGA-II) [51], [52] was introduced as a revised successor to the original NSGA [46]. In NSGA-II, elitism is ensured by combining the best parents with the best offspring obtained in every generation.

Strength Pareto evolutionary algorithm (SPEA) introduced by Zitzler and Thiele [53] use the Pareto elitism by storing nondominated solutions in an externally maintained archive. The fitness of an individual depends on the number of solutions that dominates it. For each individual in the external set, a strength value is computed. This strength is proportional to the number of solutions a certain individual dominates. The fitness of each member of the current population is then computed as the sum of the strengths of all external nondominated solutions that dominate it. The mechanism of such a fitness assignment mechanism automatically penalizes crowded solution regions and serves as a mechanism of encouraging diversity in the Pareto set without the need to specifying other parameters (such as those related to the fitness sharing mechanism). A revised version of SPEA is also introduced, referred to as SPEA2 (see [54]). The revised version incorporates a fine-grained fitness assignment strategy, which takes into account, for each individual, the number of individuals that it dominates and the number of individuals by which it is dominated. It uses a nearest neighbor density estimation technique, which guides the search more efficiently, and it has an enhanced archive truncation method that guarantees the preservation of boundary solutions.

In [55], Fieldsend *et al.* point out a problem with using an elite archive of fixed size. It is shown that limiting the size of the elite archive can produce "retreating" or "oscillating" estimates of the Pareto front. This happens as the archive is truncated when its size exceeds the limit and then new solutions are added which would have been dominated by solutions eliminated during the truncation process. Therefore, the authors recommend keeping all nondominated solutions found during the evolutionary search. To speedup processing of a large number of nondominated individuals, a tree data structure is introduced for fast searches, additions, and deletions to the archive.

In this paper, we adopt a fitness assignment that is similar to the one introduced in SPEA2 [54]. Also, as is the case for a number of previously proposed multiobjective evolutionary approaches, we maintain an external elitist archive of continuously updated Pareto solutions. The size of the external archive is not fixed and the truncation procedure suggested in SPEA2 is not used. In our implementation, we use a mechanism that ensures that boundary solutions (best solutions in each objective) are always selected as parents for the next generation. This technique was suggested by Fieldsend *et al.* [55] and was found to be effective in improving the efficiency of MO-GART.

III. Multiobjective Evolutionary ART Architectures

MO-GART uses a multiobjective evolutionary approach to find networks that achieve Pareto-optimal performance in terms of two objectives: maximizing classification accuracy and minimizing complexity (size) of ARTMAP classifier. Formally, MO-GART tries to find a set of networks (solutions) that optimizes $f(x)$ of two objectives, where

$$f(x_i) = [f_1(x_i), f_2(x_i)]^T \tag{1}$$

where f_1 represents the classification error of the network, f_2 represents the complexity of the network, and x_i represents a feasible solution or network in the space of all possible feasible networks (F) that can be applied to a specific classification task. MO-GART therefore tries to find the set of Pareto-optimal solutions (networks) that minimizes the two objectives, f_1 and f_2.

Solution $x_i \in F$ is said to be nondominated if there exist no other solution $x \in F$ such that

$$\forall\, j : f_j(x) \le f_j(x_i), \qquad j = 1, 2, \qquad \exists\, j : f_j(x) < f_j(x_i). \tag{2}$$

A Pareto-optimal solution in our case is a network such that there is no other network that has a smaller classification error using a smaller number of categories. The entire set of such optimal tradeoff solutions is the Pareto front of ART networks.

$P(0) \leftarrow$ Generate-Initial-Population();
$A(0) \leftarrow$ Initialize-Empty-Archive();
for $t \leftarrow 1$ **to** Gen_{max} **do**
  Evaluation();
  Update-Archive($P(t)$, $A(t)$);
  if *stopping criteria met* **then** exit for;
  $P'(t) \leftarrow$ Selection($P(t)$, $A(t)$);
  $P(t) \leftarrow$ Reproduction($P'(t)$);
end
return $A(t)$;

Fig. 4. Pseudocode of MO-GART algorithm.

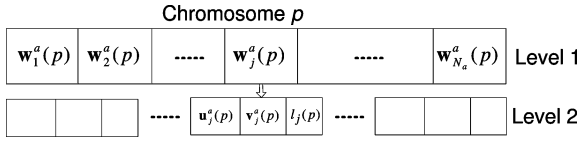

Fig. 5. MO-GFAM chromosome structure. At level 2, the category's weight $\mathbf{w}_j^a$ contains the information about the lower endpoint $\mathbf{u}_j^a$ and the upper endpoint $\mathbf{v}_j^a$ of the hyperbox corresponding to the category, as well as the label l_j of the category.

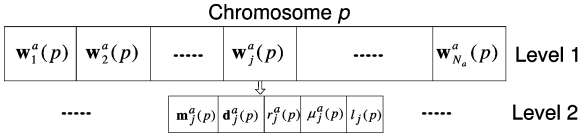

Fig. 6. MO-GEAM chromosome structure. At level 2, the category's weight $\mathbf{w}_j^a$ contains the information of the center $\mathbf{m}_j^a$, the direction vector of the major axis $\mathbf{d}_j^a$, the radius (half length) of the major axis r_j^a, and the ratio of the lengths of the minor axes over the length of the major axis μ_j^a of the ellipsoid corresponding to this category, as well as the label l_j of the category.

MO-GART operates by applying, repeatedly, genetic operators on an initial population of trained ART networks. The following pseudocode shows the basic steps of MO-GART.

The algorithm starts by generating an initial population $P(0)$ of ARTMAP networks (FAM, EAM, or GAM), each one of them trained with a different value of the baseline vigilance parameter $\bar{\rho}_a$, and order of training pattern presentation. We varied the baseline vigilance parameter between the values of 0.1 and 0.95 for FAM and EAM, and between 0 and 0.5 for GAM. The choice parameter in a FAM network was chosen to be equal to 0.01. The choice parameter in EAM network was chosen to be equal to 0.001. The initial value of the standard deviation γ in a GAM network was chosen to be equal to 0.6. The training is performed using a random subset of the available data, referred to as the training set. In our implementation, we fixed the population size $\mathrm{Pop}_{\mathrm{size}} = 20$. The networks are encoded into chromosomes, where each component (gene) represents a category (hidden node) of an ART network. Each component contains the weight information for the category. The chromosomes in MO-GART are variable length, where the length is equal to the number of categories in the network represented by the chromosome (see Figs. 5–7).

Also, MO-GART initializes an empty secondary population $A(0)$ that will be used to store nondominated solutions found

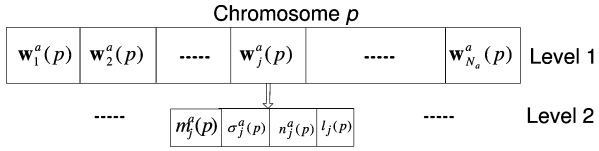

Fig. 7. MO-GGAM chromosome structure. At level 2, the category's weight $\mathbf{w}_j^a$ contains the information of the center of the Gaussian curve $\mathbf{m}_j^a$, the standard deviation vector of the Gaussian curve σ_j^a, and the number of points represented by the Gaussian curve n_j^a as well as the label l_j of the category.

during the evolution. In each generation, each solution in the population is evaluated according to each objective function. That is, the error rate of each ARTMAP network is evaluated by running it against a subset of the available data, referred to as the validation set. The second objective, complexity, is represented by the number of categories present in each network. Once networks in population P are evaluated, the archive A is updated by adding to it the solutions in P that are nondominated by solutions in A. Also, solutions in A that are now dominated by solutions just added from P are removed from the archive A. This mechanism ensures elitism. We do not impose an upper limit for the size of A since in our case (optimization of ART networks in terms of accuracy and size) the number of nondominated solutions is not expected to be too large, as the different network sizes are not that many for the problems that we experimented with.

The algorithm runs for a maximum number of generations defined by $\mathrm{Gen}_{\max}$. In our implementation, we set $\mathrm{Gen}_{\max} = 500$. However, to avoid running MO-GART for unnecessarily large number of generations, the evolution is also stopped when the archive A is not updated for ten consecutive generations.

The selection process creates a temporary population P', where the parent chromosomes used to create the next generation are selected. The chromosomes in the archive A and population P are assigned fitness values based on dominance relationship, as suggested in SPEA2 [54]. In this scheme, each individual is assigned a strength value equal to the number of solutions it dominates. After that, a raw fitness $R(x)$ is assigned for each individual as the sum of the strengths of all its dominators in both A and P. The raw fitness is then adjusted as follows. For each individual x, the distance, in objective space, to the kth nearest neighbor is found and denoted as $\sigma_k(x)$. The value of k is chosen to be the square root of the sum of the size of the archive and population. The fitness of each individual is then calculated using the following equation:

$$\mathrm{Fit}(x) = R(x) + \frac{1}{\sigma_k + 2}. \tag{3}$$

The parents are then chosen using a deterministic binary tournament selection with replacement, as follows. For each parent, randomly select two chromosomes from the combined set of A and P, and choose the chromosome with the smallest fitness value. Boundary solutions, which are networks with smallest error rate and smallest size, are ensured to be copied in the set of parents.

Once the selection step determines the parents, reproduction operators are used to create individuals for the next generation. The two well-known operators for reproduction in GAs are

TABLE I
DATABASES USED IN EXPERIMENTATION

Database Name	Total Instances	Training Instances	Validation Instances	Test Instances	Attributes	Classes	Majority
1Ci/Sq	10000	2000	5000	3000	2	2	50.00%
G4c-25	10500	500	5000	5000	2	4	25.00%
G6c-15	10512	504	5004	5004	2	6	16.67%
Iris	10100	500	4800	4800	2	2	50.00%
PAGE	5473	500	2486	2487	10	5	89.80%
Pendigits	10992	4494	3000	3498	16	10	10.00%
Sat	6436	2000	2436	2000	36	6	24.19%
Seg	2310	800	810	700	19	7	14.29%
wav	5000	1000	2000	2000	21	3	33.33%
Abalone	4177	501	1838	1838	7	3	33.30%
Optdigits	5620	1823	2000	1797	64	10	10.00%

crossover and mutation. In this work, in addition to crossover, two mutation-based operators are proposed. The first is referred to as the *mutation operator*, and it performs Gaussian mutations on the weights of the categories of the ARTMAP network. The second operator, referred to as the *prune operator*, prunes a network by deleting a number of categories from that network (structural mutation).

To avoid the need for finding proper values for the mutation and pruning probabilities, or setting default values that might result in suboptimal operation, an adaptation mechanism was employed to automatically adjust, based on performance, the invocation of reproduction operators. This performance-based adaptation is implemented at the gene (category) level. More specifically, adaptive, performance-based, parameters are computed for each component in the individual. The performance feedback relies on a metric defined for each category, referred to as the *confidence factor* (CF; see [56]). The confidence factor is a metric that measures the performance at the category level. The performance of a category is defined in terms of its accuracy and relative frequency of selection

$$\mathrm{CF}_j^k(p) = 0.5A_j^k(p) + 0.5S_j^k(p) \tag{4}$$

where $A_j^k(p)$ is the accuracy of classification achieved by category j, in the pth network, that is mapped to label k, relative to the best accuracy achieved by any category in the same network that is mapped to the same label. Furthermore, $S_j^k(p)$ is the probability of selection of category j in the pth network that is mapped to label k, relative to the maximally selected category in the same network that is mapped to the same label. In other words, $S_j^k(p)$ is the ratio of the number of times that category j is chosen over the number of times that the maximally selected category that mapped to the same label k was chosen. The scaling ensures that $A_j^k(p) \in [0,1]$, $S_j^k(p) \in [0,1]$, and therefore, $\mathrm{CF}_j(p) \in [0,1]$.

Once CF is calculated for each category, we delete categories from every chromosome in the temporary generation $P'(t)$, with probability of $1 - \mathrm{CF}_j^k(p)$. Also, the weights of every category are mutated using a Gaussian distribution that has a mean of 0 and a small standard deviation that is proportional to $(1 - \mathrm{CF}_j^k(p))$. Therefore, categories with low CF are more likely to be eliminated or more severely mutated. The applied mutation is described below.

- In MO-GFAM, for each category, either its u or v endpoint is selected randomly (with 50% probability) and then every component of this selected vector gets mutated by adding to it a small number drawn from a Gaussian distribution with zero mean. If the component of the selected vector becomes smaller than 0 or greater than 1 (after mutation), it is set back to 0 or 1, respectively.
- In MO-GEAM, for each category, every component of the ellipsoidal center m gets mutated by adding to it a small number. This number is drawn from a Gaussian distribution with zero mean. Furthermore, the mutated category's axis ratio μ or radius r is selected with 50% probability and mutated by adding to it a number from a Gaussian distribution with zero mean. If the component of the selected vector becomes smaller than 0 or greater than 1 (after mutation), it is set back to 0 or 1, respectively.
- In MO-GAM, for each category, either its mean vector m or standard deviation vector σ is selected randomly (50% probability) and mutated by adding to it a number drawn from a Gaussian distribution with zero mean. If the component of the selected vector becomes smaller than 0 or greater than 1 (after mutation), it is set back to 0 or 1, respectively.

The standard deviation of the Gaussian distribution used for the mutation of the vector components described above is proportional to $(1 - \mathrm{CF}_j^k(p))$. It is clear that $(1 - \mathrm{CF}_j^k(p)) \in [0,1]$. Also the mutated vectors components described above are in the range $[0,1]$. Since about 99% of the samples drawn from Gaussian distribution will be in the range of three standard deviation, we multiply it by a small factor to ensure mutations are not destructive. Based on our experimentation a range of values work well. We choose to set the standard deviation to be $0.05(1-\mathrm{CF}_j^k(p))$. It should be noted that categories with smaller CF are mutated more severely than categories with higher CF.

The chromosomes in $P(t)$ are then replaced by chromosomes created by crossing over pairs of parents in $P'(t)$. For each parent p, p', a random crossover point is chosen, designated as

TABLE II
C-METRIC VALUES FOR MO-GART VERSUS ssART

Database Name	C(MO-GFAM, ssFAM)	C(ssFAM, MO-GFAM)	C(MO-GEAM, ssEAM)	C(ssEAM, MO-GEAM)	C(MO-GGAM, ssGAM)	C(ssGAM, MO-GGAM)
1Ci/Sq	0.550	0.380	0.954	0.158	0.941	0.053
g4c-25	1.000	0.050	1.000	0.000	0.950	0.000
g6c-15	1.000	0.000	1.000	0.000	0.900	0.000
Iris	0.500	0.233	0.900	0.183	0.871	0.245
page	1.000	0.033	1.000	0.000	1.000	0.025
pendigits	0.542	0.191	0.826	0.143	0.639	0.236
sat	1.000	0.000	0.922	0.066	0.950	0.030
seg	0.700	0.180	0.772	0.265	0.879	0.000
wav	1.000	0.000	1.000	0.000	1.000	0.000
Abalone	0.875	0.021	1.000	0.000	1.000	0.000
Optdigits	1.000	0.000	0.875	0.000	1.000	0.000

TABLE III
TOTAL RUNTIME FOR MO-GFAM, MO-GEAM, AND MO-GGAM COMPARED TO TOTAL RUNTIME FOR ssFAM, ssEAM, AND ssGAM

Database Name	MO-GFAM	ssFAM	Gain	MO-GEAM	ssEAM	Gain	MO-GGAM	ssGAM	Gain
1Ci/Sq	22.6406	216.42	9.56	63.6581	1167.67	18.34	39.4752	1431.35	36.26
G4C-25	4.4593	130.92	29.36	11.4466	916.78	80.09	9.1047	314.49	34.54
G6C-15	5.6252	145.16	25.80	12.4733	508.22	40.74	9.8514	266.77	27.08
Iris	1.4202	21.30	15.00	4.7829	123.56	25.83	7.4811	109.25	14.60
page	4.2141	69.52	16.50	8.497	484.15	56.98	8.7017	125.68	14.44
pendigits	462.578	7864.27	17.00	997.23	58865.05	59.03	129.90	20050.39	154.35
sat	170.1563	2034.29	11.96	382.3469	17162.45	44.89	84.8639	4302.16	50.69
seg	10.2047	85.55	8.38	41.0937	1331.47	32.40	7.4577	509.99	68.38
wav	23.103	763.99	33.07	68.2825	8612.65	126.13	8.3548	1199.58	143.58
Abalone	3.80	62.16	16.36	10.50	1380.22	131.45	4.36	179.21	41.10
Optdigits	209.10	3824.18	18.23	1363.37	38828.72	28.48	131.42	8980.52	68.33

n, n', respectively. Then, all the categories with index greater than n' in the chromosome p' and all the categories with index less than or equal to index n in the chromosome with index p are moved into an empty chromosome within the new generation.

As mentioned above, the evolutionary process continues until one (of the two) stopping criterion is triggered. MO-GART does not return a single trained ART classifier, but rather, a number of ART classifiers that were present in the archive A at the last generation of the evolutionary process. These classifiers have achieved varying levels of accuracy and complexity. These alternatives are then presented to the user to make a final decision of choosing one (or more) of these classifiers. For example, if the user is mostly interested in accuracy, then the network that produced the best accuracy is chosen.

Based on our exposition above, in MO-GART, three GA parameters are set to default values and used for every data set that we experimented with: 1) the population size $(= 20)$; 2) an upper limit on the number of evolutions allowed $(= 500)$; and 3) a constant value $(= 0.05)$ used in defining the mutation probability. As a result, with MO-GART, the user is freed from the task of experimenting with parameter values that are difficult to prespecify, and quite often problem dependent.

IV. EXPERIMENTAL RESULTS

A. Preliminaries

In this section, we compare MO-GART's performance to that of other competitive ART architectures, as well as two other popular classifiers: SVM [57] and CART [21]. The measures of comparison are accuracy of the classifier, size of the classifier, and complexity of the classifier. It is worth pointing out that in the comparison of MO-GART's performance with other ART classifiers and SVM, CART is fair since we have used the same data to design, validate, and test all of these models.

We have experimented with 11 databases (see Table I), of which three are simulated databases and eight are real databases. Each database was randomly divided into three subsets: training, validation, and testing. The simulated databases include two Gaussian databases: G4C-25 and G6C-15. These are 2-D databases with four classes and six classes, and 15% and 25% overlap, between the classes. The overlap in these Gaussian data sets implies that if the optimal (Bayes) classifier were to be used the classification error attained (optimal error) would be equal to the overlap percentage. The database denoted by 1Ci/Sq is the benchmark one circle in a square problem,

TABLE IV
C-METRIC VALUES FOR MO-GART VERSUS GART [42]

Database Name	C(MO-GFAM, GFAM)	C(GFAM, MO-GFAM)	C(MO-GEAM, GEAM)	C(GEAM, MO-GEAM)	C(MO-GGAM, GGAM)	C(GGAM, MO-GGAM)
1Ci/Sq	0.482	0.429	0.417	0.382	0.467	0.362
G4C-25	0.600	0.333	0.550	0.400	0.600	0.542
G6C-15	0.500	0.300	0.700	0.408	0.750	0.408
Iris	0.700	0.150	0.700	0.267	0.600	0.148
page	0.550	0.533	0.717	0.242	0.550	0.408
pendigits	0.350	0.270	0.235	0.218	0.300	0.196
sat	0.592	0.134	0.417	0.199	0.433	0.176
seg	0.348	0.325	0.443	0.341	0.525	0.384
wav	0.767	0.150	0.700	0.183	0.600	0.500
Abalone	0.840	0.325	0.880	0.292	0.720	0.428
Optdigits	0.880	0.020	1.000	0.000	0.960	0.013

TABLE V
TOTAL RUNTIME FOR MO-GFAM, MO-GEAM, AND MO-GGAM COMPARED TO TOTAL RUNTIME FOR GFAM, GEAM, AND GGAM

Database	MO-GFAM	GFAM	Gain	MO-GEAM	GEAM	Gain	MO-GGAM	GGAM	Gain
1Ci/Sq	22.64	68.51	3.03	63.66	152.13	2.39	39.48	142.29	3.60
G4C-25	4.46	52.59	11.79	11.45	82.99	7.25	9.10	89.32	9.81
G6C-15	5.63	92.01	16.36	12.47	127.15	10.19	9.85	137.82	13.99
Iris	1.42	13.48	9.49	4.78	21.56	4.51	7.48	22.02	2.94
page	4.21	52.97	12.57	8.49	60.91	7.17	8.70	75.03	8.62
pendigits	462.57	1142.43	2.47	997.23	4304.84	4.32	129.90	537.62	4.14
sat	170.16	508.21	2.99	382.35	1234.62	3.23	84.86	329.07	3.88
seg	10.20	41.93	4.11	41.09	88.45	2.15	7.46	64.13	8.60
wav	23.10	147.23	6.37	68.28	369.38	5.41	8.35	83.94	10.05
Abalone	3.80	53.07	13.97	10.50	81.06	7.72	4.36	59.73	4.46
Optdigits	209.10	437.97	2.09	1363.37	1825.37	1.33	131.42	167.68	1.28

2-D, and two-class classification problem. The probability of finding a data point within a circle or inside the square of the circle is equal to 1/2. The rest of the databases were obtained from the University of California at Irvine (UCI) repository (see [58]) and they include: Modified Iris, Page Blocks (PAGE), Pendigits, Satellite Image (SAT), Image Segmentation (SEG), Waveform (WAV), Abalone, and Optdigits. More details about these databases can be found there.

Each database is randomly divided into a training set, a validation test, and a test set. The training set is used for the training of ART architectures under consideration. The validation set is used to estimate the classification error during the evolutionary process as explained above. Finally, the test set is used to assess the performance of the optimized networks created.

B. Comparison With Other ART Architectures

In this section, we compare MO-GART's performance to that of other competitive ART architectures, which have been proposed in the literature with the intent of addressing the category proliferation problem, such as ssFAM, ssEAM, and ssGAM. These approaches are based on the principle of semisupervision [20]. Semisupervision is a term attributed to learning in an ART architecture (FAM, EAM, or GAM), where categories in ART are allowed to encode patterns of different labels provided that the percentage of patterns that belong to the plurality label exceed a certain threshold. We also compare the performance of MO-GART to the previously introduced [42] GART architectures that did not employ a multiobjective evolutionary approach. In the GART case, the Pareto front is produced by varying the weight in the fitness function.

Since, in this work, we are not only focusing on generalization performance, but also on the size (complexity) of the network produced, it becomes more complicated to compare and rank networks. To provide a fair comparison, we resort to a comparison approach that considers the two objectives simultaneously. Since the existence of the two, sometimes competing, objectives result in multiple good solutions rather than one "best" solution, in our comparison, we assess multiple solutions (sets of solutions) produced by the different algorithms, under consideration. In other words, for each classification algorithm, we produce a number of classifiers that have attained the two objectives (good generalization and small size) at different levels of success. Then, we choose the *nondominated solutions*. A nondominated solution is defined to be a network, where no other network achieves better generalization utilizing and equal or

TABLE VI
MOST ACCURATE NETWORKS AND THEIR SIZES: FAM

	MO-GFAM			GFAM			ssFAM		
	PCC	**Size**	**PCC-CI**	**PCC**	**Size**	**PCC-CI**	**PCC**	**Size**	**PCC-CI**
1Ci/Sq	97.97	31	(97.46, 98.47)	98.07	41	(97.57, 98.56)	98.10	78	(97.61, 98.59)
G4C-25	76.00	4	(74.82, 77.18)	74.94	4	(73.74, 76.14)	74.22	4	(73.01, 75.43)
G6C-15	84.59	6	(83.59, 85.59)	84.57	6	(83.57, 85.57)	82.49	9	(81.44, 83.55)
Iris	95.19	2	(94.58, 95.79)	94.96	2	(94.34, 95.58)	94.56	2	(93.92, 95.20)
page	96.45	5	(95.72, 97.18)	96.59	6	(95.88, 97.31)	94.77	6	(93.89, 95.64)
pendigits	98.27	271	(97.83, 98.70)	97.20	282	(96.65, 97.75)	97.14	66	(96.59, 97.69)
sat	89.12	175	(87.75, 90.48)	87.90	310	(86.47, 89.33)	84.20	51	(82.60, 85.80)
seg	95.43	25	(93.88, 96.98)	94.86	22	(93.22, 96.49)	94.14	32	(92.40, 95.88)
wav	86.30	3	(84.79, 87.81)	84.35	5	(82.76, 85.94)	75.65	16	(73.77, 77.53)
Abalone	66.50	5	(64.34, 68.66)	62.25	8	(60.03, 64.47)	56.89	34	(54.63, 59.15)
Optdigits	98.05	272	(97.44, 98.66)	91.99	42	(90.80, 93.18)	87.20	52	(85.74, 88.66)
Average	89.44			87.97			85.40		

TABLE VII
MOST ACCURATE NETWORKS AND THEIR SIZES: EAM

	MO-GEAM			GEAM			ssEAM		
	PCC	**Size**	**PCC-CI**	**PCC**	**Size**	**PCC-CI**	**PCC**	**Size**	**PCC-CI**
1Ci/Sq	99.76	2	(99.58, 99.94)	99.70	2	(99.50, 99.90)	97.40	99	(96.83, 97.97)
G4C-25	75.54	4	(74.35, 76.73)	75.14	4	(73.94, 76.34)	73.90	4	(72.68, 75.12)
G6C-15	84.69	6	(83.69, 85.69)	84.87	6	(83.88, 85.86)	83.23	24	(82.20, 84.27)
Iris	95.24	2	(94.64, 95.84)	95.04	2	(94.43, 95.66)	94.65	2	(94.01, 95.28)
page	96.40	5	(95.67, 97.13)	95.09	5	(94.24, 95.94)	94.44	24	(93.54, 95.34)
pendigits	98.90	331	(98.55, 99.25)	98.31	354	(97.89, 98.74)	96.60	179	(96.00, 97.20)
sat	88.34	198	(86.93, 89.74)	87.85	203	(86.42, 89.28)	85.50	141	(83.96, 87.04)
seg	93.86	52	(92.08, 95.64)	92.14	85	(90.15, 94.14)	91.57	83	(89.51, 93.63)
wav	86.35	5	(84.85, 87.85)	85.95	9	(84.43, 87.47)	79.80	12	(78.04, 81.56)
Abalone	66.40	6	(64.24, 68.56)	62.67	4	(60.46, 64.88)	57.42	5	(55.16, 59.68)
Optdigits	98.40	418	(97.85, 98.95)	94.10	162	(93.07, 95.13)	91.93	122	(90.74, 93.12)
Average	89.44			88.26			86.04		

smaller number of categories. Our comparison between algorithms is then based on the quality of the nondominated set produced by each algorithm. We also compare the time it takes each algorithm to produce the nondominated set of solution networks.

C. Comparison With ssART

For each algorithm, ssFAM, ssEAM, and ssGAM, and for each of the 11 databases, we performed a number of experiments with different network parameter settings. In particular, the parameter settings that we experimented with ssFAM were: baseline vigilance values ranging from 0.1 to 0.9 with step size of 0.1, choice parameter values of 0.01 and 0.1, maximum allowable mixture threshold values ranging from 0 to 0.9 with step size of 0.1, and ten different orders of pattern presentations of the training data (resulting in 1800 different parameter settings). Furthermore, the settings for ssEAM were: baseline vigilance values ranging from 0.1 to 0.9 with step size of 0.1, choice parameter values of 0.001 and 0.01, maximum allowable mixture threshold values ranging from 0 to 0.5 with step size of 0.1, minimum-axes-to-maximum-axis ratio values ranging from 0.5 to 1 with step size of 0.1, and ten different orders of pattern presentations of the training data (resulting in 6480 different parameter settings). Also, the settings for ssGAM were: baseline vigilance values ranging from 0 to 0.9 with step size of 0.1, initial standard deviation parameter ranging from 0.5 to 1 with step size of 0.1, maximum allowable mixture threshold values ranging from 0 to 0.9 with step size of 0.1, and ten different orders of pattern presentations of the training data (resulting in 6000 different parameter settings). All networks are trained using the training set for ten epochs. It should be emphasized that the parameter ranges used were determined by the authors of this paper and they reflect their experience of what are good parameter settings for these ART networks. The parameter settings were chosen to provide varying levels of accuracy and complexity in these networks. Solutions that are Pareto optimal with respect to these

TABLE VIII
MOST ACCURATE NETWORKS AND THEIR SIZES: GAM

	MO-GGAM			GGAM			ssGAM		
	PCC	**Size**	**PCC-CI**	**PCC**	**Size**	**PCC-CI**	**PCC**	**Size**	**PCC-CI**
1Ci/Sq	99.80	2	(99.64, 99.96)	99.83	2	(99.69, 99.98)	94.63	26	(93.83, 95.44)
G4C-25	75.92	4	(74.73, 77.11)	75.24	4	(74.04, 76.44)	74.84	23	(73.64, 76.04)
G6C-15	85.17	6	(84.19, 86.16)	84.97	6	(83.98, 85.96)	85.07	20	(84.08, 86.06)
Iris	94.90	2	(94.27, 95.52)	94.85	2	(94.23, 95.48)	95.21	7	(94.60, 95.81)
page	96.38	5	(95.64, 97.11)	96.34	6	(95.61, 97.08)	94.52	7	(93.62, 95.41)
pendigits	98.10	88	(97.65, 98.55)	97.83	108	(97.34, 98.31)	97.43	87	(96.90, 97.95)
sat	88.75	106	(87.37, 90.13)	88.35	118	(86.94, 89.76)	87.00	81	(85.53, 88.47)
seg	92.59	13	(90.65, 94.53)	92.71	17	(90.79, 94.64)	91.29	31	(89.20, 93.38)
wav	87.15	4	(85.68, 88.62)	87.80	4	(86.37, 89.23)	85.35	11	(83.80, 86.90)
Abalone	67.30	5	(65.16, 69.44)	61.06	5	(58.83, 63.29)	57.19	30	(54.93, 59.45)
Optdigits	97.15	161	(96.42, 97.88)	93.32	72	(92.23, 94.41)	92.21	55	(91.04, 93.38)
Average	89.38			88.39			86.79		

TABLE IX
PERFORMANCE OF SVM AND MO-GART ON SOME DATA SETS

	MO-GFAM			MO-GEAM			MO-GGAM			SVM		
	PCC	Size	Training Time	PCC	Size	Training Time	PCC	Size	Training Time	PCC	Size	Training Time
1Ci/Sq	97.97	31	22.64	99.76	2	63.66	99.80	2	39.48	99.67	88	136.41
G4C-25	76.00	4	4.46	75.54	4	11.45	75.92	4	9.10	75.24	277	42.09
G6C-15	84.59	6	5.63	84.69	6	12.47	85.17	6	9.85	84.99	504	37.93
Iris	95.19	2	1.42	95.24	2	4.78	94.90	2	7.48	95.04	79	20.04
page	96.45	5	4.21	96.40	5	8.50	95.38	5	8.70	95.30	150	20.49
pendigits	98.27	271	462.58	98.90	331	997.23	98.1	88	129.90	99.54	929	616.88
sat	89.12	175	170.16	88.34	198	382.35	88.75	106	84.86	90.25	1081	263.78
seg	95.43	25	10.20	92.86	52	41.09	92.59	13	7.46	97.29	230	27.28
wav	86.30	3	23.10	86.35	5	68.28	87.15	4	8.35	87.45	574	74.28
Abalone	66.50	5	3.80	66.40	6	10.50	67.30	5	4.36	61.66	337	41.05
Optdigits	98.0	272	209.10	98.4	418	1363.37	97.15	161	131.42	97.22	673	474.48

two objective were finally chosen. Pareto optimality is determined using the validation set. The total computation time required to obtain these network solutions for each database and each method, which is the sum of training and validation times in seconds for all the parameter settings, is reported in Table III, and referred to as the *total runtime*.

A one-to-one comparison of the results reported in Table III reveals that the *total runtime* of the MO-GART networks is, in all cases, between one and two orders of magnitude smaller than the *total runtime* of their corresponding counterparts, ssART networks. To compare the quality of the solutions produced by MO-GFAM versus ssFAM, MO-GEAM versus ssEAM, and finally, MO-GGAM versus ssGAM, we use a metric that compares the network solutions obtained by the semisupervised ART network (for all different parameter settings) and the ones obtained by the MO-GART architectures. This metric has been used before in similar situations (see [53]) for multiobjective optimization problems, and is defined as follows:

$$C(A,B) = \frac{|b \in B : \exists\, a \in A, b \prec a|}{|B|}. \tag{5}$$

This metric measures the fraction of members in set B that are dominated by at least one member in set A. Therefore, $C(A,B) = 1$ means all members in B are dominated by members in A. In this case, the approach that produced set A is a clear winner. It is obvious that we need to consider also $C(B,A)$ in order to properly compare the two sets.

Since the calculated values of $C(A,B)$ and $C(B,A)$ are dependent on the random seed used to evolve the population of FAMs, EAMs, and GAMs in the MO-GART approach, we produced network solutions by changing the seed ten times. Consequently, ten different values of the C metric were produced for each comparison pair. In Table II, we compare the average values (over the ten replications) of C(MO-GFAM, ssFAM) versus C(ssFAM, MO-GFAM), C(MO-GEAM, ssEAM) versus C(ssEAM, MO-GEAM), and C(MO-GGAM, ssGAM) versus C(ssGAM, MO-GGAM) . It is obvious from the table that the average values of C(MO-GFAM, ssFAM) are larger than C(ssFAM, MO-GFAM) values, which indicates that networks produced by MO-GFAM are more likely to dominate networks produced by ssFAM, and therefore, the networks produced

TABLE X
PERFORMANCE OF CART AND MO-GART ON SOME DATA SETS

	MO-GFAM			MO-GEAM			MO-GGAM			CART		
	PCC	Size	Training Time	PCC	Size	Training Time	PCC	Size	Training Time	PCC	Size	Training Time
1Ci/Sq	97.97	31	22.64	99.76	2	63.66	99.80	2	39.48	97.57	28	0.02
G4C-25	76.00	4	4.46	75.54	4	11.45	75.92	4	9.10	73.50	4	0.00
G6C-15	84.59	6	5.63	84.69	6	12.47	85.17	6	9.85	80.42	6	0.02
glass	76.56	6	0.13	75.31	6	0.21	76.00	8	0.28	64.06	4	0.00
Iris	95.19	2	1.42	95.24	2	4.78	94.90	2	7.48	94.02	4	0.00
page	96.45	5	4.21	96.40	5	8.50	95.38	5	8.70	93.84	7	0.02
pendigits	98.27	271	462.58	98.90	331	997.23	98.1	88	129.90	93.37	109	0.28
pima	82.67	2	0.12	83.33	4	0.41	82.67	2	0.23	73.71	2	0.00
sat	89.12	175	170.16	88.34	198	382.35	88.75	106	84.86	84.35	22	0.30
seg	95.43	25	10.20	92.86	52	41.09	92.59	13	7.46	93.43	17	0.05
wav	86.30	3	23.10	86.35	5	68.28	87.15	4	8.35	75.20	14	0.09
Abalone	66.50	5	3.80	66.40	6	10.50	67.30	5	4.36	61.18	17	0.01
Optdigits	98.0	272	209.10	98.4	418	1363.37	97.15	161	131.42	82.42	88	0.28

TABLE XI
SSART RESULT STATISTICS

	ssFAM				**ssEAM**				**ssGAM**			
Dataset	**COUNT**	**AVG**	**MAX**	**STD**	**COUNT**	**AVG**	**MAX**	**STD**	**COUNT**	**AVG**	**MAX**	**STD**
1Ci/Sq	1800	81.19	98.33	14.90	6480	86.36	97.47	9.25	6000	75.94	97.17	16.23
G4C-25	1800	63.40	74.40	9.26	6480	66.82	73.90	2.64	6000	58.34	75.10	15.63
G6C-15	1800	70.22	82.89	12.50	6480	75.93	83.23	3.66	6000	63.20	85.07	19.94
Iris	1800	89.94	95.10	6.43	6480	91.76	95.27	3.78	6000	84.60	95.29	14.38
page	1800	86.15	95.05	16.38	6480	88.71	94.98	4.04	6000	82.70	94.66	15.85
pendigits	1800	63.86	97.14	27.89	6480	81.86	97.20	20.60	6000	83.27	97.60	12.99
sat	1800	63.30	84.55	18.23	6480	80.01	86.45	4.87	6000	74.55	87.85	12.16
seg	1800	73.00	95.86	15.55	6480	83.19	93.43	5.99	6000	66.97	92.71	17.22
wav	1800	56.57	78.70	15.08	6480	65.39	79.80	17.68	6000	66.85	85.70	15.80
Abalone	1800	51.91	59.27	4.67	6480	54.70	59.39	1.46	6000	51.56	58.14	3.96
Optdigits	1800	47.25	87.70	25.78	6480	67.86	93.04	31.39	6000	68.49	92.71	19.31

by MO-GFAM are expected to be of higher quality. Similar conclusions can be drawn for MO-GEAM versus ssEAM and MO-GGAM versus ssGAM. This result is expected since MO-GART uses a multiobjective approach that is designed to produce a high-quality Pareto front networks, compared to the grid search employed for ssART. To provide a fair comparison, the performance of the most accurate networks (PCC values) is shown in Tables VI–VIII. In the same 95% confidence intervals for the attained PCC values are also depicted; the PCC value reported in these tables is the probability of correct classification on the test set produced by the network with the best PCC value attained on the validation set. As it can be easily seen, the MO-GART networks were able to consistently find more accurate networks using, in most instances, smaller network sizes. For instance, in Table VI, for the Abalone data set, MO-GFAM achieved a classification accuracy of 66.50% compared to a 56.89% classification accuracy, achieved by ssFAM, utilizing five instead of 34 categories.

D. Comparison With GART

For GART, it is not possible to produce the nondominated solutions in one run. Rather, it is necessary to run the algorithm multiple times to produce the different nondominated solutions. We chose to run GART using five different settings for the fitness weight. The weights used were 0.01, 0.05, 0.1, 0.2, and 0.5. We repeated this process ten times to account for the stochasticity of the genetic algorithm. The average time it took to produce one set of nondominated solutions is reported in Table V, and referred to as the *total runtime*. The results in Table IV show an advantage of MO-GART over GART in terms of solution quality. Also, a one-to-one comparison of the results reported in Table V reveals that the *total runtime* of the MO-GART networks is smaller than the *total runtime* of their corresponding counterparts, GART networks, and quite often an order of magnitude smaller. Tables VI–VIII compare the most accurate network (PCC values) obtained from MO-GART and

TABLE XII
PERFORMANCE OF SVM AND MO-GART ON SOME DATA SETS WITH CONFIDENCE INTERVALS

	MO-GFAM		MO-GEAM		MO-GGAM		SVM	
	PCC	**CI**	**PCC**	**CI**	**PCC**	**CI**	**PCC**	**CI**
1Ci/Sq	97.97	(97.46, 98.47)	99.76	(99.58, 99.94)	99.80	(99.64, 99.96)	99.67	(99.46, 99.87)
G4C-25	76.00	(74.82, 77.18)	75.54	(74.35, 76.73)	75.92	(74.73, 77.11)	75.24	(74.04, 76.44)
G6C-15	84.59	(83.59, 85.59)	84.69	(83.69, 85.69)	85.17	(84.19, 86.16)	84.83	(83.84, 85.83)
Iris	95.19	(94.58, 95.79)	95.24	(94.64, 95.84)	94.90	(94.27, 95.52)	95.06	(94.45, 95.68)
page	96.45	(95.72, 97.18)	96.40	(95.67, 97.13)	96.38	(95.64, 97.11)	95.30	(94.47, 96.14)
pendigits	98.27	(97.83, 98.70)	98.90	(98.55, 99.25)	98.10	(97.65, 98.55)	99.54	(99.32, 99.77)
sat	89.12	(87.75, 90.48)	88.34	(86.93, 89.74)	88.75	(87.37, 90.13)	90.25	(88.95, 91.55)
seg	95.43	(93.88, 96.98)	93.86	(92.08, 95.64)	92.59	(90.65, 94.53)	97.29	(96.08, 98.49)
wav	86.30	(84.79, 87.81)	86.35	(84.85, 87.85)	87.15	(85.68, 88.62)	87.45	(86.00, 88.90)
Abalone	66.50	(64.34, 68.66)	66.40	(64.24, 68.56)	67.30	(65.16, 69.44)	61.66	(59.44, 63.88)
Optdigits	98.05	(97.44, 98.66)	98.40	(97.85, 98.95)	97.15	(96.42, 97.88)	97.22	(96.50, 97.94)
Average	89.44		89.44		89.38		89.41	

TABLE XIII
SVM RESULTS STATISTICS

Dataset	**COUNT**	**AVG**	**MAX**	**STD**
1Ci/Sq	110	73.60	99.78	23.54
g4c-25	110	74.20	75.24	0.85
g6c-15	110	83.67	85.13	1.28
Iris	110	94.25	95.06	0.38
page	110	94.06	95.63	0.62
pendigits	110	81.99	99.57	29.90
sat	110	75.70	90.55	21.84
seg	110	72.36	97.29	31.64
wav	110	74.42	87.95	21.65
abalone	110	51.24	63.39	12.16
optdigits	110	71.06	97.38	36.24

GART; in these tables, the confidence intervals of these PCC values are depicted. The network with lowest classification error on the validation set is chosen and referred to as the most accurate network, and then the error rate using the test set is reported. As it can be seen from these tables, MO-GART was, in some cases, able to find a better solution than GART, while the reverse situation was never true. Therefore, achieving a better quality solution at a lower computational cost, in addition to producing multiple optimal solutions at once, justifies the proposed approach of using a multiobjective approach to evolve ART architectures.

E. Comparison With Other Classifiers

In this section, we compare MO-GART's performance to that of other popular classifiers. One classifier we compared MO-GART with was the SVM. This is a state-of-the-art classifier known for its excellent generalization performance, and strong theoretical foundation. For SVM training, we used LIBSVM (v. 2.8, see [59]), which is an implementation of the sequential minimal optimization (SMO) algorithm first developed by Platt [60]. As far as we are aware, this is the best-known implementation of SMO available. The SVM classifier has several adjustable parameters including C, a regularization parameter, and parameters associated with the kernel. For this test, we employed the radial basis function (RBF) kernel [or Gaussian kernel $\exp(-\gamma\|x_i - x_j\|^2)$]. The linear kernel may be more suitable in some cases where the data are known to be linearly separable and can produce fewer support vectors. However, the RBF kernel is more applicable to a wider variety of instances including linearly separable data. In general, it is not known, *a priori*, which are the best parameter settings for a particular data set for providing the best generalization performance. For this reason, it is not uncommon to train the classifier for a wide range of settings to find the optimal generalization performance. For this test, we trained the SVM classifier across a grid of C and γ parameters. The range of values used is as follows: $\log 2(C)$ was varied between -5 and 15, with step size of 2 and $\log 2(\gamma)$ was varied between 3 and -15 with step size of -2, resulting in a total number of 110 C and γ parameter values.

Another classifier we compared with was CART. CART (see [21]) is a classical methodology for training decision trees. It provides systematic and complete solutions to the key problems in training, including the selection of splits, the separation of training into growing and pruning phases to prevent overtraining, the handling of missing values, and the cross validation of trees. It also provides theoretical support for speedup algorithms and the performance evaluation of trees. During training, CART offers options and parameters to tune its performance, but it also provides default settings, such as the usage of Gini index as the impurity measure in growing the tree and the 1-SE rule (see [21] for more details) in selecting the best pruned tree; these choices usually produce good results. We applied these default settings in our experiments.

Tables IX and X show the results obtained using the most accurate SVM classifier, the CART classifier, as well as the

TABLE XIV
PERFORMANCE OF CART AND MO-GART ON SOME DATA SETS WITH CONFIDENCE INTERVALS

	MO-GFAM		MO-GEAM		MO-GGAM		CART	
	PCC	CI	PCC	CI	PCC	CI	PCC	CI
1Ci/Sq	97.97	(97.46, 98.47)	99.76	(99.58, 99.94)	99.80	(99.64, 99.96)	97.57	(97.02, 98.12)
G4C-25	76.00	(74.82, 77.18)	75.54	(74.35, 76.73)	75.92	(74.73, 77.11)	73.50	(72.28, 74.72)
G6C-15	84.59	(83.59, 85.59)	84.69	(83.69, 85.69)	85.17	(84.19, 86.16)	80.42	(79.32, 81.52)
Iris	95.19	(94.58, 95.79)	95.24	(94.64, 95.84)	94.90	(94.27, 95.52)	94.02	(93.35, 94.69)
page	96.45	(95.72, 97.18)	96.40	(95.67, 97.13)	96.38	(95.64, 97.11)	93.84	(92.89, 94.78)
pendigits	98.27	(97.83, 98.70)	98.90	(98.55, 99.25)	98.10	(97.65, 98.55)	93.37	(92.54, 94.19)
sat	89.12	(87.75, 90.48)	88.34	(86.93, 89.74)	88.75	(87.37, 90.13)	84.35	(82.76, 85.94)
seg	95.43	(93.88, 96.98)	93.86	(92.08, 95.64)	92.59	(90.65, 94.53)	93.43	(91.59, 95.26)
wav	86.30	(84.79, 87.81)	86.35	(84.85, 87.85)	87.15	(85.68, 88.62)	75.20	(73.31, 77.09)
Abalone	66.50	(64.34, 68.66)	66.40	(64.24, 68.56)	67.30	(65.16, 69.44)	61.18	(58.95, 63.41)
Optdigits	98.05	(97.44, 98.66)	98.40	(97.85, 98.95)	97.15	(96.42, 97.88)	82.42	(80.75, 84.09)
Average	89.44		89.44		89.38		84.48	

most accurate MO-GART classifiers. Comparing the results in Tables IX and X, one can make the following observations.

- The accuracy of the most accurate MO-GART classifier is very competitive with the accuracy of the most accurate SVM classifier.
- The size of the most accurate MO-GART classifier is smaller, in all instances, than the size of the most accurate SVM network, and at times, significantly smaller (e.g., MO-GFAM has three categories for the waveform data set, while SVM has 574 support vectors).
- The total runtime required to produce the most accurate MO-GART classifier is, at almost all instances, smaller than the total runtime needed to produce the most accurate SVM classifier, and at times, an order of magnitude smaller (e.g., MO-GFAM needed 1.42 s for the Iris data set, while SVM needed 20.04 s of total runtime).
- The accuracy of the most accurate MO-GART classifier is better, in all instances, than the accuracy of the most accurate CART classifier, and in some instances, a lot better (e.g., MO-GGAM accuracy for the waveform data set is 87.15%, while the CART accuracy is 75.20%).
- The size of the most accurate MO-GART classifier is competitive with the size of the most accurate CART classifier, in some instances larger, but quite often very close to the size of the most accurate CART classifier.
- The total runtime required to produce the most accurate MO-GART classifier is significantly larger than the total runtime needed to produce the most accurate CART network; an expected result.

As we mentioned earlier, the comparison of MO-GART, SVM, and CART provided above used the same databases and data sets per database for training, validation, and testing of these architectures. This comparison illustrates that the family of MO-GART classifiers, presented in this paper, is competitive in comparison with other, popular, state-of-the-art classifiers.

V. DISCUSSION OF RESULTS

A. Remarks About MO-GART and ssART Comparisons

In Table II, the quality of nondominated solutions produced by MO-GFAM is compared with the quality of the nondominated solutions produced by the ssFAM classifier on 11 different data sets, based on the C metric. In Table III, the total runtime required to train a MO-GFAM classifier and an ssFAM classifier for all the ssFAM parameter settings, reported in Section IV-C, is also shown. Furthermore, in Table VI, the most accurate MO-GFAM and ssFAM classifiers are reported (PCC values), as well as the confidence intervals associated with the reported PCC values. The obvious observation that comes out of the comparisons shown in Tables II, III, and VII is that MO-GFAM is, for most data sets, more accurate than ssFAM, and it attains this accuracy requiring less computational resources.

In Table II, the quality of nondominated solutions produced by MO-GEAM is compared with the quality of the nondominated solutions produced by the ssEAM classifier on 11 different data sets, based on the C metric. In Table III, the total runtime required to train a MO-GEAM classifier and an ssEAM classifier for all the ssEAM parameter settings, reported in Section IV-C, is also shown. Furthermore, in Table VI, the most accurate MO-GEAM and ssEAM classifiers are reported (PCC values), as well as the confidence intervals associated with the reported PCC values. The obvious observation that comes out of the comparisons shown in Tables II, III, and VI is that MO-GEAM is, for most data sets, more accurate than ssEAM, and it attains this accuracy requiring less computational resources.

In Table II, the quality of nondominated solutions produced by MO-GGAM is compared with the quality of the nondominated solutions produced by the ssGAM classifier on 11 different data sets, based on the C metric. In Table III, the total runtime required to train a MO-GGAM classifier and an ssGAM classifier for all the ssGAM parameter settings, reported in

TABLE XV
GART MOST ACCURATE NETWORK SENSITIVITY TO SEED (TEN REPLICATIONS)

	GFAM				GEAM				GGAM			
	MIN	**MAX**	**AVG**	**STD**	**MIN**	**MAX**	**AVG**	**STD**	**MIN**	**MAX**	**AVG**	**STD**
1Ci/Sq	96.83	98.07	97.30	0.23	97.43	99.70	99.54	0.78	96.37	99.83	97.16	1.02
g4c-25	74.56	74.94	74.69	0.12	74.40	75.14	74.80	0.24	74.92	75.24	75.06	0.11
g6c-15	84.07	84.57	84.39	0.25	84.41	84.87	84.72	0.16	84.51	84.97	84.80	0.14
Iris	94.42	94.96	94.77	0.28	93.81	95.04	94.68	0.36	94.73	94.85	94.79	0.04
page	95.70	96.59	96.25	0.29	94.09	95.09	94.75	0.36	95.23	96.34	95.72	0.38
pendigits	96.57	97.20	97.02	0.13	96.77	98.31	97.76	0.49	97.20	97.83	97.54	0.20
sat	86.40	87.90	87.11	0.75	85.40	87.85	86.60	0.70	85.25	88.35	87.57	0.91
seg	93.14	94.86	94.16	0.91	89.71	92.14	92.60	1.21	88.57	92.71	90.33	1.13
wav	80.20	84.35	83.05	1.70	84.05	85.95	85.14	0.82	85.75	87.80	86.95	0.56
Abalone	60.17	62.25	61.41	0.75	59.87	62.67	61.19	1.02	60.17	61.06	61.00	0.56
Optdigits	89.09	91.99	90.44	1.04	84.53	94.10	88.49	3.09	91.10	93.32	92.70	0.99

TABLE XVI
MO-GART MOST ACCURATE NETWORK SENSITIVITY TO SEED (TEN REPLICATIONS)

	MO-GFAM				MO-GEAM				MO-GGAM			
	MIN	**MAX**	**AVG**	**STD**	**MIN**	**MAX**	**AVG**	**STD**	**MIN**	**MAX**	**AVG**	**STD**
1Ci/Sq	97.72	97.97	97.87	0.11	97.66	99.76	98.66	0.95	97.00	99.80	97.85	1.00
g4c-25	74.60	76.00	75.56	0.50	74.48	75.52	75.16	0.45	74.72	75.92	75.54	0.51
g6c-15	83.71	84.59	84.29	0.37	84.49	84.69	84.56	0.11	84.59	85.17	84.80	0.17
Iris	94.98	95.19	95.14	0.11	94.65	95.24	94.90	0.15	94.85	94.90	94.87	0.02
page	95.23	96.45	95.78	0.40	95.99	96.40	96.14	0.21	94.48	96.38	95.75	0.95
pendigits	97.94	98.27	98.09	0.17	98.14	98.90	98.52	0.41	97.82	98.10	97.95	0.15
sat	88.25	89.12	88.86	0.36	87.05	88.34	87.96	0.48	87.60	88.75	88.24	0.61
seg	94.00	95.43	94.73	0.50	92.35	93.86	92.89	0.40	91.51	92.59	92.35	0.48
wav	82.15	86.30	83.89	1.14	84.75	86.35	85.53	0.44	86.80	87.15	86.92	0.18
Abalone	65.95	66.50	66.19	0.29	65.42	66.40	65.67	0.45	67.12	67.30	67.18	0.09
Optdigits	97.78	98.05	97.93	0.14	97.95	98.40	98.23	0.24	96.98	97.15	97.09	0.09

Section IV-C , is also shown. Furthermore, in Table VIII, the most accurate MO-GGAM and ssGAM classifiers are reported (PCC values), as well as the confidence intervals associated with the reported PCC values. The obvious observation that comes out of the comparisons shown in Tables II, III, and VIII is that MO-GGAM is, for a number of data sets, more accurate than ssGAM, and it attains this accuracy requiring less computational resources.

One could argue we do not need to experiment with as many ssFAM parameters as the ones that we have experimented with. To counteract this argument, we show in Table XI the maximum, mean, and standard deviation of the accuracy attained by ssFAM (PCC on the test set) for all the different parameter values (1800 of them) that ssFAM was trained and validated with. It is obvious from Table XI that there is a difference in all of these table entries between the average and maximum performances of an ssFAM network for a particular data set, as the ssFAM network parameters change. For instance, for the Pendigits data set, we observe an average, maximum, and standard deviation of the generalization performance (over the 1800 ssFAM parameter values that we experimented with) of 63.86, 97.14, and 27.89, respectively. As we have mentioned earlier, unless the ssFAM user has a lot of experience experimenting with ssFAM, it is hard to limit the amount of parameter experimentation needed to produce an accurate ssFAM network for a specific data set. MO-GFAM releases the user from the arduous task of guessing and experimenting with a variety of ssFAM parameter values to discover a good network; with MO-GFAM, the user does not need to specify any of the ssFAM parameter values but instead the user only specifies default parameter values for the initial GA population (20), number of generations (500), and a constant parameter (0.05) that affects the mutation probability.

Similar results as the ones provided in Table IX about the dependence of ssFAM generalization performance on the network parameter values are also depicted, in the same table, for ssEAM, as well as for ssGAM. Similar observations as the ones that we derived for ssFAM hold for the dependence of ssEAM's and ssGAM's generalization performances on the network parameter values. In a nutshell, ssFAM's, ssEAM's, and ssGAM's generalization performances are strongly dependent on the choice of the network parameter values, and as a

result, extensive experimentation is most often needed to produce good performing ssFAM, ssEAM, or ssGAM classifiers.

B. Remarks About MO-GART and GART Comparisons

In Table IV, the quality of nondominated solutions produced by MO-GFAM is compared with the quality of the nondominated solutions produced by the GFAM classifier on 11 different data sets, based on the C metric. In Table IV, the total runtimes required to train a MO-GFAM classifier and a GFAM classifier (for all the different sets of weights of the GFAM fitness function, reported in Section IV-D) are also shown. Furthermore, in Table V, the most accurate MO-GFAM and GFAM classifiers are reported (PCC values), as well as the confidence intervals associated with the reported PCC values. The obvious observation that comes out of the comparisons shown in Tables IV–VI is that in a couple of data sets MO-GFAM is more accurate than GFAM (the reverse statement is never true), and it attains this accuracy requiring less computational resources.

In Table IV, the quality of nondominated solutions produced by MO-GEAM is compared with the quality of the nondominated solutions produced by the GEAM classifier on 11 different data sets, based on the C metric. In Table V, the total runtimes required to train a MO-GEAM classifier and a GEAM classifier (for all the different sets of weights of the GFAM fitness function, reported in Section IV-D) are also shown. Furthermore, in Table VII, the most accurate MO-GEAM and GEAM classifiers are reported (PCC values), as well as the confidence intervals associated with the reported PCC values. The obvious observation that comes out of the comparisons shown in Tables IV–VI is that in a few data sets MO-GEAM is more accurate than GEAM (the reverse statement is never true), and it attains this accuracy requiring less computational resources.

In Table IV, the quality of nondominated solutions produced by MO-GGAM is compared with the quality of the nondominated solutions produced by the GGAM classifier on 11 different data sets, based on the C metric. In Table V, the total runtimes required to train a MO-GGAM classifier and a GGAM classifier (for all the different sets of weights of the GFAM fitness function, reported in Section IV-D) are also shown. Furthermore, in Table VIII, the most accurate MO-GGAM and GGAM classifiers are reported (PCC values), as well as the confidence intervals associated with the reported PCC values. The obvious observation that comes out of the comparisons shown in Tables IV, V, and VIII is that in some data sets MO-GGAM is more accurate than GGAM (the reverse statement is never true), and it attains this accuracy requiring less computational resources.

C. Remarks About MO-GART and SVM Comparisons

In Table IX, the performance of MO-GART is compared with the performance of SVM classifiers on 11 different data sets. The results depicted in Table IX show the performance (PCC on the test set) of the most accurate (PCC on the validation set) SVM and MO-GART classifiers. Furthermore, they show the size of this most accurate SVM and MO-GART classifier, as well as the time required to produce such accurate SVM and MO-GART classifiers. From the results in Table IX, we observe that MO-GART has 1) produced a classifier of accuracy (PCC) slightly less than the accuracy of the SVM classifier (in most instances), 2) produced a classifier whose size is in all instances smaller than the corresponding size of the SVM classifier, and 3) produced a classifier by requiring computational resources that are less (sometimes much less) than the computational resources required to produce the corresponding SVM classifier.

In Table XII, we are showing the accuracy comparisons (PCC on the test set) between MO-GART and SVM for the 11 data sets, but for each such comparison, we are also including the confidence interval associated with the PCC numbers, reported in Table IX. These confidence intervals validate our claim that the SVM generalization performance is slightly higher than the MO-GART performance on a couple of the 11 data sets, MO-GART performance is higher on a couple of other data sets, and in most cases, it cannot be determined which classifier is best.

To produce the SVM PCC number of Table IX, we trained and validated the SVM classifier for a number of the C and γ parameters (the specific C and γ parameter values are mentioned in Section IV-E). Table XIII shows the maximum, mean, and standard deviation of the SVM PCC values obtained by experimenting with all these different values of C and γ. The purpose of Table XIII is to provide a snapshot for the reader of how important it is to train and validate the SVM classifier on a number of parameter values in order to attain the maximum SVM PCC performance (depicted in Table XIII). It is obvious from Table XIII that this experimentation with the different C and γ values is needed. For instance, the average, maximum, and standard deviation of the PCC values pertaining to the Optdigits data set are 71.06, 97.38, and 36.24, respectively, indicating the large variability of the Optdigits data set's generalization performance with respect to the change of the parameter values (C and γ); a number of other data sets in Table XI also exhibit wide variability of generalization performance with respect to the change of the C and γ parameter values.

Given an arbitrary data set, and having chosen a kernel (in our SVM experiments, we chose the very popular RBF kernel), the optimal parameter values (C and γ) are usually unknown *a priori*. As a result, a grid search across a range of parameter values is required. Furthermore, without prior knowledge or experience with a particular type of data set, it can be difficult to attempt to narrow the range of search values. In this work, we chose a range of values for our grid search based upon that reported in the literature [61], for the RBF kernel, which is also reported in much of the SVM literature. In particular, the $\log 2(C)$ value changed from -15 to 2 with step size of 2, and the $\log 2(\gamma)$ value changed from 3 to -15 with step size of -2. As it can be seen from Table XI, there can be a significant amount of performance variation across the range of parameter values, and each data set responds differently, justifying the need for such an extensive experimentation.

D. Remarks About the MO-GART and CART Comparisons

In Table X, we have provided comparisons between the most accurate MO-GART and CART classifiers on 11 data sets. One obvious observation from these comparisons is that the training time required by the CART classifier is negligible, and much

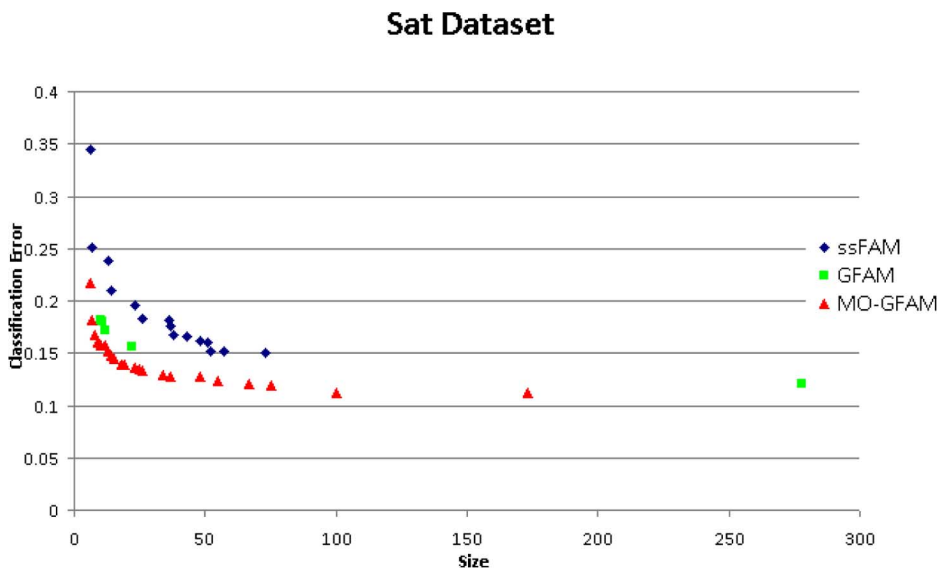

Fig. 8. Pareto fronts for the satellite database.

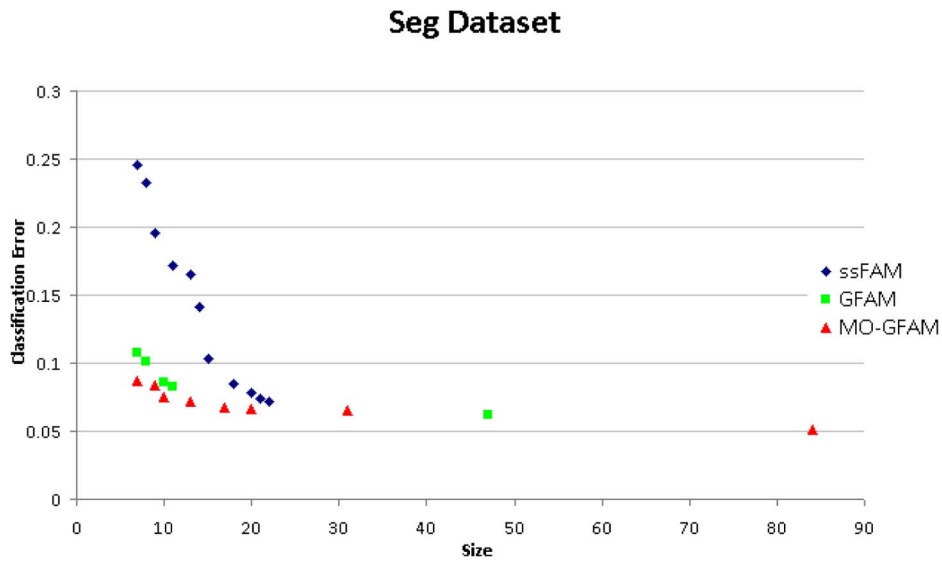

Fig. 9. Pareto front for the segmentation database.

smaller than the training time required by the MO-GART classifier. Another observation is that the size needed by the CART classifier is comparable with the size needed by the MO-GART classifier that attains the highest accuracy (although differences exist for some of the data sets). The last observation is that the most accurate MO-GART classifier is more accurate (in all instances) than the corresponding CART classifier.

In Table XIV, we are showing the accuracy comparisons between MO-GART and CART for the 11 data sets, but for each such comparison, we are also including the confidence interval associated with the PCC numbers, reported in Table X. These confidence intervals validate our claim that the MO-GART generalization performance is superior to the CART performance on all of the 11 data sets, except one.

E. Remarks About the Sensitivity of MO-GART to the Seed Value

We have emphasized above that the performance of ssART and SVM classifiers is sensitive to the choice of their respective parameter values. On the contrary, in MO-GART, we start with certain default parameter values (population size of 20, number of generations needed of 500, and a mutation probability parameter constant of 0.05) that stay fixed as experimentation moves from one data set to the next. Also, with MO-GART, we do not require the user to specify the ART parameter values, or the SVM parameter values, a tough task for someone who is a novice with these types of classifiers. The only value that changes in MO-GART is the seed value that specifies the initial population of trained ART architectures that is subsequently evolved to produce a nondominated archive of solutions at the time when MO-GART convergence is achieved. In Table XIV, the average, maximum, and standard deviation of the PCC (on the test set) corresponding to the best performing PCC (on the validation set) MO-GART network from the archive at the last generation of the evolutionary process, are shown. The results illustrate a good feature of the MO-GART classifier that is the low variability of its generalization performance with respect to the initial seed value, in defining contrast of the high variability

of the ssART and SVM classifiers with respect to the parameter values needed to be specified for each one of these classifier paradigms. Actually, this advantage of MO-GART (low variability of its generalization performance with respect to the seed value) is also shared by GART, as well (see illustrative results in Table XV).

F. Pareto Fronts for Some Data Sets

In Section IV, we compared the quality of the solutions provided by MO-GART, GART, and ssART using a popular metric, referred to as the C metric. Other metrics of comparison exist between the sets of solutions provided by MO-GART, GART, and ssART, as reported in [53], such as the S metric. Due to the extensive volume of the results provided so far in Sections IV and V and the strong conclusions, derived from these results regarding the merits of MO-GART, compared to other competitive ART classifiers, as well as other popular classifier models (e.g., SVM and CART), we refrain from comparing the solutions sets, obtained by MO-GART, GART, and ssART, using other than the C metric. However, it is instructive to provide a sample of the Pareto fronts, produced by MO-GART, GART, and ssART on the data sets that we experimented with in this effort (e.g., see Figs. 8 and 9 where the Pareto fronts produced by MO-GFAM, GFAM, and ssFAM for the Satellite, and Segmentation data sets are depicted). These figures provide a visual reaffirmation of the advantages of the MO-GART approach compared to the GART and ssART approaches of producing a good classifier model.

The better generalization results produced by MO-GART (see Figs. 8 and 9) for an illustration of this fact) compared to the single-objective approach of GART could be attributed to the fact that using fixed weights for the two objectives is not enough of producing as good and/or as diverse solutions as MO-GART, despite the fact that multiple sets of fixed weights were used with GART. It is possible that the single-objective GART approach could produce better family of solutions, if the weights of the single-objective approach are allowed to change during the evolution (see [8]), but this investigation was beyond the scope of our paper.

VI. Summary and Conclusion

In this paper, we introduce, for the first time, a multiobjective evolutionary approach to optimize ARTMAP neural networks in terms of two objectives: classification accuracy (higher is better) and classifier complexity (smaller is better). In particular, we apply a MOEA to optimize the performance of three well-known ART architectures: fuzzy ARTMAP, ellipsoidal ARTMAP, and Gaussian ARTMAP. The resulting architectures are referred to as MO-GFAM, MO-GEAM, and MO-GGAM, and collectively as MO-GART.

The MO-GART approach presents a solution to the category proliferation problem in ART. Other approaches to solve the category proliferation problem in ART have been proposed before, such as the semisupervised ART (ss-ART) approach (ssFAM, ssEAM, and ssGAM). An extensive comparison of MO-GART and the ss-ART approach concluded that the MO-GART approach is more elegant (does not require tweaking of the ART network parameters), more effective (produces higher accuracy and smaller size network solutions), and more efficient (faster) than the ss-ART approach. The results, presented in Tables II and III, indicate that MO-GART offers clear advantages compared to ss-ART; it is worth noting that ss-ART is a class of well-performing ART classifiers that compares very favorably with other ART and non-ART classifiers. The advantage of MO-GART compared to GART (a related approach to evolve ART networks) is that MO-GART focuses on two objectives at once. Consequently, MO-GART does not require multiple GA runs to produce multiple good solutions to the classification problem, under consideration, and hence it is more efficient than the GART approach (as Table V reveals). Finally, MO-GART is more elegant than GART because it does not require user intervention to specify *a priori* the preference towards one objective (accuracy) versus the other (size). In closing, it was also shown that the family of MO-GART classifiers is competitive compared to other popular, state-of-the-art classifiers that have appeared in the literature, such as SVM and CART.

References

[1] S. Grossberg, "Adaptive pattern classification and universal recoding, II: Feedback, expectation, olfaction, and illusions," *Biol. Cybern.*, vol. 23, no. 4, pp. 187–202, Dec. 1976.

[2] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 698–713, Sep. 1992.

[3] G. Anagnostopoulos, "Novel approaches in adaptive resonance theory for machine learning," Ph.D. dissertation, Dept. Electr. Comput. Eng., Univ. Central Florida, Orlando, FL, 2001.

[4] J. R. Williamson, "Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps," *Neural Netw.*, vol. 9, no. 5, pp. 881–897, 1996.

[5] A. Al-Daraiseh, A. Kaylani, M. Georgiopoulos, A. S. Wu, M. Mollaghasemi, and G. C. Anagnostopoulos, "GFAM: Evolving fuzzy ARTMAP neural networks," *Neural Netw.*, vol. 20, no. 8, pp. 873–891, Oct. 2007.

[6] A. Kaylani, A. Al-Daraiseh, M. Georgiopoulos, M. Mollaghasemi, G. C. Anagnostopoulos, and A. S. Wu, "Genetic optimization of ART neural network architectures," in *Proc. Int. Joint Conf. Neural Netw.*, Aug. 2007, pp. 1114–1119.

[7] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[8] Y. Jin, T. Okabe, and B. Sendhoff, "Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how?," in *Proc. Genetic Evol. Comput. Conf.*, L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, Eds., San Francisco, CA, 2001, pp. 1042–1049.

[9] Y. Jin, T. Okabe, and B. Sendhoff, "Neural network regularization and ensembling using multi-objective evolutionary algorithms," in *Proc. Congr. Evol. Comput.*, 2004, DOI: 10.1109/CEC.2004.1330830.

[10] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Construction of fuzzy classification systems with rectangular fuzzy rules using genetic algorithms," *Fuzzy Sets Syst.*, vol. 65, no. 2/3, pp. 237–253, 1994.

[11] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy-if-then rules for classification problems using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 3, pp. 260–270, Aug. 1995.

[12] H. Ishibuchi, T. Murata, and I. B. Turksen, "Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems," *Fuzzy Sets Syst.*, vol. 89, no. 2, pp. 135–150, 1997.

[13] H. Ishibuchi, T. Nakashima, and T. Murata, "Three objective genetics based machine learning for linguistic rule extraction," in *Proc. Evol. Multi-Objective Optim.*, 2001, pp. 588–602.

[14] H. Ishibuchi, T. Nakashima, and T. Murata, "Three objective genetics based machine learning for linguistic rule extraction," *Inf. Sci.*, vol. 136, no. 1–4, pp. 109–133, 2001.

[15] H. Ishibuchi and Y. Nojima, "Multi-objective optimization in linguistic rule extraction from numerical data," *Int. J. Approx. Reason.*, vol. 44, pp. 4–31, 2007.
[16] R. M. Everson and J. E. Fieldsend, "Multi-class ROC analysis from a multi-objective optimization perspective," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 918–927, 2006.
[17] Y. Jin, R. Wen, and B. Sendhoff, "Evolutionary multi-objective optimization of spiking neural networks," in *Proceedings of the International Conference of Artificial Neural Networks (ICANN)*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer-Verlag, 2007, vol. 4668, pp. 370–379.
[18] L. Graning, Y. Jin, and B. Sendhoff, "Generalization improvement in multi-objective learning," in *Proc. Int. Joint Conf. Neural Netw.*, 2006, pp. 9983–9900.
[19] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 38, no. 3, pp. 397–415, May 2008.
[20] G. C. Anagnostopoulos, M. Bharadwaj, M. Georgiopoulos, S. J. Verzi, and G. L. Heileman, "Exemplar-based pattern recognition via semi-supervised learning," in *Proc. Int. Joint Conf. Neural Netw.*, Portland, OR, 2003, vol. 4, pp. 2782–2787.
[21] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees* . New York: Chapman & Hall, 1984.
[22] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Comput. Vis. Graph. Image Process.*, vol. 37, pp. 54–115, 1987.
[23] G. A. Carpenter and S. Grossberg, "ART 2: Self-organization of stable category recognition codes for analog input patterns," *Appl. Opt.*, vol. 26, no. 23, 1987.
[24] G. A. Carpenter, S. Grossberg, and J. H. Reynorlds, "ARTMAP: Supervised real-time learning and classification of non-stationary data by a self-organizing neural network," *Neural Netw.*, vol. 6, pp. 565–588, 1991.
[25] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Netw.*, vol. 4, no. 6, pp. 759–771, 1991.
[26] S. Marriott and R. F. Harrison, "A modified fuzzy ARTMAP architecture for the approximation of noisy mappings," *Neural Netw.*, vol. 8, pp. 619–641, 1995.
[27] E. Gomez-Sanchez, Y. Dimitriadis, J. Cano-Izquierdo, and J. Lopez-Coronado, "MicroARTMAP: Use of mutual information for category reduction in Fuzzy ARTMAP," *IEEE Trans. Neural Netw.*, vol. 13, no. 1, pp. 1045–9227, Jan. 2002.
[28] E. Gomez-Sanchez, Y. Dimitriadis, J. Cano-Izquierdo, and J. Lopez-Coronado, "Safe-μ ARTMAP: A new solution for reducing category proliferation in Fuzzy ARTMAP," in *Proc. Int. Joint Conf. Neural Netw.*, Washington, DC, 2001, vol. 2, pp. 1197–1202.
[29] A. Koufakou, M. Georgiopoulos, G. Anagnostopoulos, and T. Kasparis, "Cross-validation in Fuzzy ARTMAP for large databases," *Neural Netw.*, vol. 14, pp. 1279–1291, 2001.
[30] G. A. Carpenter, B. L. Milenova, and B. W. Noeske, "Distributed ARTMAP: A neural network for fast distributed supervised learning," *Neural Netw.*, vol. 11, no. 5, pp. 793–813, 1998.
[31] J. R. Williamson, "A constructive, incremental-learning network for mixture modeling and classification," *Neural Comput.*, vol. 9, no. 7, pp. 1517–1543, 1997.
[32] E. Parrado-Hernadez, E. Gomez-Sanchez, and Y. Dimitriadis, "Study of distributed learning as a solution to category proliferation in Fuzzy ARTMAP based neural systems," *Neural Netw.*, vol. 16, pp. 1039–1057.
[33] K. A. D. Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Comput. Commun. Sci., Univ. Michigan, Ann Arbor, MI, 1975.
[34] H. Muhlenbein and D. Schlierkamp-Voosen, "Predictive models for the breeder genetic algorithm I: Continuous parameter optimization," *Evol. Comput.*, vol. 1, no. 1, pp. 25–49, 1993.
[35] T. Back, "Self-adaptation in genetic algorithms," in *Proc. 1st Eur. Conf. Artif. Life* , Washington, DC, 1992, pp. 263–271.
[36] R. Hinterding, Z. Michalewicz, and A. E. Eiben, "Adaptation in evolutionary computation: A survey," in *Proc. IEEE Int. Conf. Evol. Comput./IEEE World Congr. Comput. Intell.*, I. Alekser and E. J. Taylor, Eds., Amsterdam, The Netherlands, Apr. 1997, pp. 65–69.
[37] T. C. Fogarty, "Varying the probability of mutation in the genetic algorithm," in *Proc. 3rd Int. Conf. Genetic Algorithms* , San Francisco, CA, 1989, pp. 104–109.
[38] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, "Combining convergence and diversity in evolutionary multiobjective optimization," *Evol. Comput.*, vol. 10, no. 3, pp. 263–282, 2002.
[39] P. J. Angeline, G. M. Saunders, and J. P. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 54–65, Jan. 1994.
[40] N. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Trans. Syst. Man Cybern.*, vol. 24, no. 2, pp. 656–667, Apr. 1994.
[41] C. C. Coello, "Evolutionary multi-objective optimization: A historical view of the field," *IEEE Comput. Intell. Mag.*, vol. 1, no. 1, pp. 28–36, Feb. 2006.
[42] A. Kaylani, M. Georgiopoulos, M. Mollaghasemi, and G. Anagnostopoulos, "Genetic optimization of ART neural network architectures," in *Proc. Artif. Intell. Soft Comput. Conf.*, 2007, pp. 225–230.
[43] R. Santos, J. Nievola, and A. Freitas, "Extracting comprehensible rules from neural networks via genetic algorithms," in *Proc. IEEE Symp. Combinat. Evol. Comput. Neural Netw.*, 2000, pp. 130–139.
[44] I. Das and J. Dennis, "A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems," *Structural Optim.*, vol. 14, pp. 63–69, 1997.
[45] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms* , Mahwah, NJ, 1985, pp. 93–100.
[46] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1994.
[47] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proc. 5th Int. Conf. Genetic Algorithms* , San Mateo, CA, 1993, pp. 416–423.
[48] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Proc. 1st IEEE Conf. Evol. Comput./IEEE World Congr. Comput. Intell.*, Piscataway, NJ, 1994, vol. 1, pp. 82–87.
[49] M. Erickson, A. Mayer, and J. Horn, "The niched Pareto genetic algorithm 2 applied to the design of groundwater remediation systems," in *Proc. 1st Int. Conf. Evol. Multi-Criterion Optim.* , London, U.K., 2001, pp. 681–695.
[50] H. Ishibuchi and T. Murata, "Multi-objective genetic local search algorithm," in *Proc. IEEE Int. Conf. Evol. Comput.*, May 1996, pp. 119–124.
[51] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Proc. Parallel Problem Solving Nature VI Conf.*, Paris, France, 2000, pp. 849–858.
[52] K. Deb, A. Pratab, S. Agrawal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[53] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.
[54] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," Comput. Eng. Netw. Lab. (TIK), Swiss Federal Inst. Technol. (ETH), Zurich, Switzerland, Tech. Rep. 103, 2001.
[55] J. E. Fieldsend, R. M. Everson, and S. Singh, "Using unconstrained elite archives for multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 7, no. 3, pp. 305–323, Jun. 2003.
[56] G. A. Carpenter and H. A. Tan, "Rule extraction: From neural architecture to symbolic representation," *Connection Sci.*, vol. 7, pp. 3–27, 1995.
[57] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowl. Disc.*, vol. 2, no. 2, pp. 121–167, Jun. 1998.
[58] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz, UCI Repository of Machine Learning Databases, Univ. California at Irvine, Irvine, CA, 1998 [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html
[59] C.-C. Chang and C.-J. Lin, LIBSVM: A Library for Support Vector Machines, Nat. Taiwan Univ., Taiwan, 2001 [Online]. Available: http://www.csie.ntu.edu.tw/cjlin/libsvm
[60] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods: Support Vector Learning*, B. Schölkopf, C. J. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 185–208.

[61] R. Fan, P. Chen, and C. J. Lin, "Working set selection using second order information for training SVM," *J. Mach. Learn. Res.*, vol. 6, pp. 1889–1918, 2005.

Assem Kaylani received the B.S. degree in electrical engineering from the University of Jordan in 1998, the M.S. degree in computer engineering and the Certificate Degree in systems simulation for engineers from the University of Central Florida, Orlando, in 2001 and 2005, respectively, and the Ph.D. degree in computer engineering focusing on data mining algorithms as a research area in 2008.

He is currently a Research and Development Manager at Dubai-based Incube, FZCO, where he is leading the development and integration of enterprise software designed to automate and optimize the different activities related to supply chains, logistics, and distribution. During 2003–2008, he worked as a Senior Software Engineer at Florida-based Productivity Apex, Inc., where he led the software development of several NASA funded projects including shuttle ground operations simulation modeling and the generic simulation environment for modeling future launch operations (GEM-FLO). In addition, he was involved in several data mining projects performed for Lockheed, Siemens Westinghouse, Boeing, and others.

Michael Georgiopoulos received the Diploma in electrical engineering from the National Technical University, Athens, Greece, in 1981 and the M.S. degree and Ph.D. degree in electrical engineering from the University of Connecticut, Storrs, in 1983 and 1983, respectively.

He is currently a Professor in the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando. His research interests lie in the areas of machine learning and applications with special emphasis on neural network and neuroevolutionary algorithms, and their applications. He has published more than 60 journal papers and more than 180 conference papers in a variety of conference and journal venues.

Dr. Georgiopoulos has been an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS from 2002 to 2006, and he is currently serving as an Associate Editor of the *Neural Networks* journal. He served as the General Chair of the S+SSPR 2008, the satellite workshop affiliated with the 2008 International Conference on Pattern Recognition. He is currently serving as the Technical Co-Chair of the 2011 International Joint Conference on Neural Networks.

Mansooreh Mollaghasemi received the B.S. degree and the M.S. degree in chemical engineering and the Ph.D. degree in industrial engineering from the University of Louisville, Louisville, KY, in 1991.

She is an Associate Professor at the Department of Industrial Engineering and Management Systems, University of Central Florida, Orlando. Her area of expertise includes multiple criteria decision making, simulation modeling and analysis, and process optimization. She has been involved in conducting funded research through NASA, Department of Transportation, Lockheed Martin, Siemens, SAIC, Lucent Technologies, Boeing, and the U.S. Army among others. She has published in prestigious scholarly journals such as *IIE Transactions*, *Interfaces*, *Annals of Operation Research, Computers and OR*, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, *Transactions of the Society for Computer Simulation*, and the IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT. Her publications also include a book entitled *Making Multiple Objective Decisions* (Piscataway, NJ: IEEE Computer Society Press).

Georgios C. Anagnostopoulos received the Diploma in electrical engineering from the University of Patras, Patras, Greece, in 1994 and the M.S. and Ph.D. degrees in electrical engineering from the University of Central Florida, Orlando, in 1997 and 2001, respectively.

He is currently an Associate Professor of Electrical and Computer Engineering at Florida Institute of Technology, Melbourne. His research interests span machine learning with an emphasis in pattern recognition, artificial neural networks, and kernel methods.

Christopher Sentelle received the B.S. and M.S. degrees in electrical engineering from the University of Nebraska-Lincoln, Lincoln, in 1993 and 1995, respectively. He is currently working towards the Ph.D. degree in electrical engineering from the University of Central Florida, Orlando.

He is a Radar Algorithm Engineer for an Orlando, FL engineering firm. His research interests are in the area of machine learning, genetic algorithms, image processing algorithms, fuzzy methods, numerical optimization, and support vector machines. He is currently researching the application of numerical optimization techniques for fast training of SVMs for large data sets. He is also published for his work in FD-TLM electromagnetic simulation for nonlinear Josephson junction devices, image processing techniques for digital mammography, as well as in the area of machine learning.

Mr. Sentelle is a member of the HKN engineering society.

Mingyu Zhong received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2002 and the M.S. degree and the Ph.D. degree in electrical engineering from the University of Central Florida, Orlando, in 2005 and 2007, respectively.

He is currently working in AdSense as a Software Engineer at Google Inc., Kirkland, WA.