MO-GART: Multiobjective Genetic ART Architectures

A. Kaylani, M. Georgiopoulos, M. Mollaghasemi, and G. C. Anagnostopoulos

Abstract— **In this work we present, for the first time, the evolution of ART Neural Network architectures (classifiers) using a multiobjective optimization approach. In particular, we propose the use of a multiobjective evolutionary approach to evolve simultaneously the weights, as well as 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, or 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.**

I. Introduction

THE 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 on-line 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 that has been repeatedly reported in the literature is the category proliferation problem. This refers to the problem where ART, in the process of solving a classification problem, creates unnecessarily large architectures. This problem is more amplified when the data in the classification problem are noisy, and/or significantly overlapping. Another limitation of these ART architectures is the dependence of their performance on the parameters chosen in the training phase (e.g., vigilance parameter, choice parameter, order of training pattern presentation). Good choices for these parameters is problem dependent, thus requiring experimentation with various parameter choices (an expensive proposition) in order to obtain the best possibly performing ART network.

To alleviate these problems, we introduced genetic Fuzzy ARTMAP (GFAM) in [5]. GFAM uses a genetic algorithm (GA) (see [6]) to evolve simultaneously the weights, as well as the topology of FAM neural networks. In [7], we extended these ideas to EAM and GAM, and introduced several improvements to the GFAM evolutionary approach, which resulted in significant gains in terms of efficiency. Furthermore, in [7] we adopted an adaptive evolutionary approach that eliminated the dependence of performance on algorithm parameter settings. The resulting architectures were referred to as GFAM, GEAM and GGAM, and collectively as GART.

GART starts with a population of trained ART networks, whose number of nodes in the hidden layer and the values of the interconnection weights converging to these nodes are fully determined (at the beginning of the evolution) by ARTs 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 set up 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 limitation, the user should run the algorithm for different settings of the weights in the fitness function; an expensive proposition. Furthermore, the weighted sum approach might *not* be able to produce all possible solutions that might be of interest to the user (for more details see Section II). Since genetic algorithms are population-based approaches, they are suitable for finding multiple solutions if an appropriate multiobjective evolutionary algorithm (MOEA) is used.

The organization of the paper is as follows: In the next section we present a brief review of the different MOEA's introduced in the literature. In section III we discuss the proposed MO-GART algorithm. In section IV we evaluate the performance of MO-GART and compare it with GART and three other ART architectures: ssFAM, ssEAM and ssGAM (see [8]). Finally, in section V, we summarize our findings.

II. Multiobjective Evolutionary Algorithms

Many real world problems involve simultaneous optimization of conflicting objectives. This is the basic challenge

Michael Georgiopoulos is with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA (phone: 407-823-5338; fax: 407-823-5835; email: michaelg@mail.ucf.edu).

978-1-4244-1823-7/08/$25.00

of multiobjective optimization research. Evolutionary algorithms have been used extensively to solve multiobjective optimization problems, resulting in a body of knowledge known as multiobjective evolutionary algorithms (MOEA). A number of authors have published surveys of MOEA, such as [9], and the reader can find more details about MOEA's 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 GA's. The selection operator (in single 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 straight forward 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 $fit(x) = \sum_{i=1}^{L} w_i f_i(x)$ where L denotes the number of objectives and $f_i(x)$ denotes the i-th objective function. This is the approach that was adopted in GART [7] and a number of other evolutionary Neural Networks such as [10]. It follows immediately that the solution that optimizes $fit(x)$ 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 non-convex regions of the Pareto front for any combination of the weights. This has been pointed out by a number of researchers, such as [11]. 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 becomes more important when the fitness function is used to determine the selection probability, in which case the probability of selection will vary across the Pareto front solutions based on the shape of the Pareto front. This results in a poor diversity of the Pareto solutions found. As these drawbacks were identified in GART [7], in this paper we look for a multiobjective approaches that overcome these drawbacks.

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 before hand. 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 [12] where Vector Evaluated Genetic Algorithm (VEGA) was introduced. In VEGA, the selection step generates a number of sub-populations by performing proportional selection according to each objective in turn. Then these sub-populations are combined to obtain a new population, on which the genetic operators, crossover and mutation, are applied. VEGA has a major drawback which is that its selection scheme is biased towards some Pareto optimal solutions. To overcome problems identified in VEGA, [6] 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 [13] who introduced the Non-dominated Sorting Genetic Algorithm (NSGA) and [14] who introduced the Multiobjective

Genetic Algorithm (MOGA). In [15] the authors use a tournament selection scheme based on Pareto dominance in their implementation of Niched-Pareto Genetic Algorithm (NPGA). In a similar fashion, as a successor to NPGA, [16] introduced a revised version referred to as Niched Pareto Genetic Algorithm 2 (NPGA 2).

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, in [17], the authors use 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 Non-dominated Sorting Genetic Algorithm II (NSGA-II) [18], [19] was introduced as a revised successor to the original NSGA [13]. In NSGA-II elitism is ensured by combining the best parents with the best offspring obtained in every generation.

SPEA (Strength Pareto Evolutionary Algorithm) introduced by Zitzler and Thiele in [20], 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 non-dominated 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). In [21] a revised version of SPEA is introduced, referred to as SPEA2. The revised version incorporates a fine-grained fitness assignment strategy which takes into account for each individual the number of individuals that dominate it 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 [22] the authors points out to the problem of 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 that might be dominated by solutions that were eliminated during the truncation process. Therefore, the authors recommend keeping all nondominated solutions found during the evolutionary search. To speedup processing of large number of nondominated individuals, a tree data structure is introduced and used for fast searches, additions and deletion to the archive.

In this paper we adopt a fitness assignment that is similar to the one introduced in SPEA II [21]. 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 SPEA II is not used. The evolution of ART networks does not produce a large number of Pareto solutions, and therefore, the performance of the MOEA is not expected to be affected by a large archive size. 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 in [22] and was found to be effective in improving the efficiency of MO-GART. The next section describes in more details the operation of the resulting neural network architecture.

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

$P(0) \leftarrow$ 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 $A(t)$;

Algorithm 1: Pseudo Code of MO-GART Algorithm

The main difference between GART [7] and MO-GART is the selection operator. In GART the selection operator bases the selection of parents using a fitness function that combines the two objectives, using a weighted sum. In MO-GART selection is based on Pareto optimality and therefore MO-GART is capable of finding multiple solutions on the Pareto front, in one run. Also, MO-GART uses a continuously updated Pareto archive where the Pareto solutions found so far are stored.

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. In our implementation we fixed the population size, $Pop_{size} = 20$. The networks are encoded into chromosomes, where each component (gene) represents a category (hidden node) of an ART network. Each component contains the

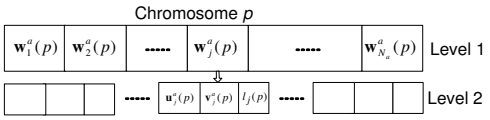

Fig. 1. MO-GFAM chromosome structure. At level 2, the category's weight $\mathbf{w}_j^a$ contains the information about the lower end-point, $\mathbf{u}_j^a$, and the upper end-point, $\mathbf{v}_j^a$, of the hyperbox corresponding to the category, as well as the label l_j of the category.

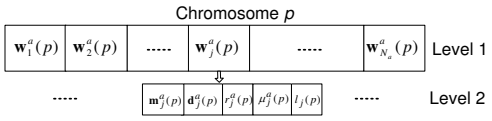

Fig. 2. MO-GEAM chromosome structure. At level 2, the category's weight $\mathbf{w}_j^a$ contains the information of the center, $\mathbf{m}_j^a$, the direction vector of the major axis, $\mathbf{d}_j^a$, the radius (half length) of the major axis, r_j^a, and the ratio of the lengths of the minor axes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

weight information for the category. The chromosomes in MO-GART (and GART) are variable length, where the length is equal to the number of categories in the network represented by the chromosome (see Figures 1, 2, 3).

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

The algorithm runs for a maximum number of generations defined by Gen_{max}. In our implementation we set $Gen_{max} = 500$. 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 10 consecutive generations.

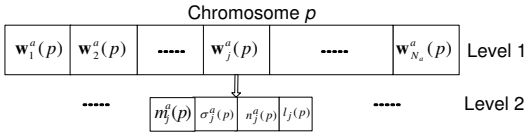

Fig. 3. MO-GGAM chromosome structure. At level 2, the category's weight $\mathbf{w}_j^a$ contains the information of the center of the Gaussian curve, $\mathbf{m}_j^a$, the standard deviation vector of the Gaussian curve, σ_j^a, and the number of points represented by the Gaussian curve, n_j^a, as well as the label l_j of the category.

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 SPEA II [21]. In this scheme each individual is assigned a strength value that is equal to the number of solutions it dominates. After that, a raw fitness, $R(x)$, is assigned for each individual to be 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 k-th 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:

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

More details about this fitness assignment can be found in [21]. 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 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 [23]). 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.

$$CF_j^k(p) = 0.5A_j^k(p) + 0.5S_j^k(p) \tag{2}$$

where $A_j^k(p)$ is the accuracy of classification achieved by category j, in the $p-th$ 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 $p-th$ network, that is mapped to label k, relative to the maximally selected category in the same network that is mapped to the same label.

Once CF is calculated for each category, with probability of $1-CF_j^k(p)$, we delete categories from every chromosome in the temporary generation $P'(t)$. Also, the weights of every category are mutated using a Gaussian distribution that has a mean of 0 and standard deviation of $0.05(1 - CF_j^k(p))$. Therefore, categories with low CF are more likely to be eliminated or more severely mutated.

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 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 and 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.

IV. Results

In this section we compare MO-GART's performance to that of other popular ART architectures. The objective of this comparison is not to compare different multiobjective evolutionary approaches; rather, the objective here is to compare the accuracy and size of several neural network architectures against the one proposed. In particular, we compare MO-GART's performance to that of other popular 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 semi-supervision [8]. Semi-supervision 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 [7] genetic ART architectures (GART) that did not use a multiobjective evolutionary approach. In the GART case, the Pareto front is produced by varying the weight in the fitness function.

We have experimented with 11 databases, of which 4 are simulated databases and 7 are real databases. Each database was randomly divided into three subsets; training, validation and testing. The simulated databases include 2 Gaussian databases: G4C-25 and G6C-15. These are , 2-dimensional databases with 4-classes and 6-classes, and 15% and 25% overlap, between the classes. The database denoted by 1Ci/Sq is the benchmark one circle in a square problem, 2-dimensional, two class classification problem. The probability of finding a data point within a circle or inside the square and outside the circle is equal to 1/2. The rest of the databases were obtained from the UCI repository (see [24]) and they include: Modified Iris, Page Blocks (PAGE), Pendigits, Satellite Image (SAT), Image Segmentation (SEG), Waveform (WAV), Glass Identification (GLASS), and Pima-Indian Diabetes (PIMA). More details about these databases can be found there.

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 *non-dominated solutions*. A non-dominated solution is defined to be a network, where no other network achieves better generalization utilizing equal or smaller number of categories. Our comparison between algorithms is then based on the quality of the non-dominated set that was produced by each algorithm. We also compare the time it takes each algorithm to produce the non-dominated set of solution networks.

Experiments were conducted for the three MO-GART architectures: MO-GFAM, MO-GEAM, and MO-GGAM for each of the 11 databases. The average computation time in seconds (over 10 replications) needed to produce the solutions, is referred to as the *Total Run Time*, and reported in Tables II and IV.

A. Comparison with ssART

For each of the ssFAM, ssEAM, and ssGAM, and for each of the 11 databases, we performed a number of experiments with different settings of their network parameter values. In particular, we experimented with 1,800 different parameter settings for ssFAM, 6,480 different parameter settings for ssEAM and 6,000 different parameter settings for ssGAM. 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 those two objective were finally chosen. The total computation time required to obtain these network solutions for each database and each method, which is the sum of training and validation CPU times in seconds for all the tried settings, is reported in Table II, and referred to as the *Total Run Time*.

A one-to-one comparison of the results reported in Table II reveals that the *Total Run Time* of the MO-GART networks is smaller, in most instances an order of magnitude smaller than the *Total Run Time* of their corresponding counterparts, ssART networks. To compare the generalization performance of MO-GFAM and ssFAM, MO-GEAM and ssEAM, and finally MO-GGAM and ssGAM we use a metric that compares the network solutions obtained by the ss-network (for all different parameter settings) and the network solutions obtained by the MO-GART. This metric has been used before in similar situations (see [20]). This metric is defined as follows:

$$C(A,B) = \frac{|b \in B : \exists a \in A, b \prec a|}{|B|} \quad (3)$$

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 seed used to evolve the population of FAMs, EAMs and GAMs in the MO-GART approach, we produced network solutions by changing the seed 10 times. Consequently, 10 different values of the C metric were produced for each comparison pair. In Table I we compare the average values (over the 10 replications) of *C(MO-GFAM, ssFAM)* versus *C(ssFAM, MO-GFAM)*, and *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 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. To provide a fair comparison, the performance of the most accurate networks is shown in Tables V, VI and VII. As it can be easily seen, the MO-GART networks were able to consistently find more accurate networks using, in most instances, much smaller network sizes.

B. Comparison with GART

For GART (introduced in [7]) 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. We repeated this process 10 times to account for the stochasticity of the genetic algorithm. The average time it took to produce 1 set of nondominated solutions is reported in Table IV, and referred to as the *Total Run Time*. The result in Table III shows an advantage of MO-GART over GART in terms of solution quality. Also, a one-to-one comparison of the results reported in Table IV reveals that the *Total Run Time* of the MO-GART networks is smaller than the *Total Run Time* of their corresponding counterparts, GART networks. Tables V, VI and VII compares the most accurate network obtained from MO-GART and GART. As it can be seen, MO-GART in most cases was able to find a better solution than GART. Therefore, achieving better solution quality at a lower computational cost, in addition to producing multiple optimal solutions at once; justifying the proposed approach of using a multiobjective approach to evolve ART architectures.

TABLE V
MOST ACCURATE NETWORKS AND THEIR SIZES: FAM

	MO-GFAM		GFAM		ssFAM	
	PCC	**Size**	**PCC**	**Size**	**PCC**	**Size**
1Ci/Sq	97.97	31	98.07	41	98.10	78
G4C-25	76.00	4	74.94	4	74.22	4
G6C-15	84.59	6	84.57	6	82.49	9
glass	76.56	6	76.56	6	73.44	7
Iris	95.19	2	94.96	2	94.56	2
page	96.45	5	95.59	6	94.77	6
pendigits	98.27	271	97.20	282	97.14	66
pima	82.67	2	79.31	4	73.28	4
sat	89.12	175	87.90	310	84.20	51
seg	95.43	25	94.86	22	94.14	32
wav	86.30	3	84.35	5	75.65	16

TABLE VI
MOST ACCURATE NETWORKS AND THEIR SIZES: EAM

	MO-GEAM		GEAM		ssEAM	
	PCC	**Size**	**PCC**	**Size**	**PCC**	**Size**
1Ci/Sq	99.76	2	99.70	2	97.40	99
G4C-25	75.54	4	75.14	4	73.90	4
G6C-15	84.69	6	84.87	6	83.23	24
glass	75.31	6	75.00	6	73.44	17
Iris	95.24	2	95.04	2	94.65	2
page	96.40	5	95.09	5	94.44	24
pendigits	98.90	331	98.31	354	96.60	179
pima	83.33	4	78.88	3	75.00	6
sat	88.34	198	87.85	203	85.50	141
seg	93.86	52	92.14	85	91.57	83
wav	86.35	5	85.95	9	79.80	12

V. CONCLUSION

In this paper we introduce, for the first time, a multi-objective 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.

TABLE I

C-METRIC VALUES FOR MO-GART VS. 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
glass	0.533	0.342	0.956	0.058	0.871	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
pima	1.000	0.000	0.780	0.050	0.983	0.150
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

TABLE II

TOTAL RUN TIME FOR MO-GFAM, MO-GEAM AND MO-GGAM COMPARED TO TOTAL RUN TIME FOR SSFAM, SSEAM, 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
glass	0.1281	2.82	21.98	0.2092	5.72	27.34	0.2812	3.52	12.50
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
pima	0.1173	3.68	31.41	0.4063	75.88	186.76	0.2265	32.75	144.57
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

TABLE III

C-METRIC VALUES FOR MO-GART VS. GART [7]

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
glass	0.700	0.333	0.675	0.350	0.733	0.358
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
pima	1.000	0.200	0.550	0.400	0.783	0.450
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

TABLE IV

TOTAL RUN TIME FOR MO-GFAM, MO-GEAM AND MO-GGAM COMPARED TO TOTAL RUN TIME FOR GFAM, GEAM, GGAM

Database	MO-GFAM	GFAM	Gain	MO-GEAM	GEAM	Gain	MO-GGAM	GGAM	Gain
1Ci/Sq	22.6406	68.51	3.03	63.6581	152.13	2.39	39.4752	142.29	3.60
G4C-25	4.4593	52.59	11.79	11.4466	82.99	7.25	9.1047	89.32	9.81
G6C-15	5.6252	92.01	16.36	12.4733	127.15	10.19	9.8514	137.82	13.99
glass	0.1281	1.71	13.32	0.2092	2.26	10.81	0.2812	2.64	9.38
Iris	1.4202	13.48	9.49	4.7829	21.56	4.51	7.4811	22.02	2.94
page	4.2141	52.97	12.57	8.497	60.91	7.17	8.7017	75.03	8.62
pendigits	462.578	1142.43	2.47	997.23	4304.84	4.32	129.90	537.62	4.14
pima	0.1173	0.95	8.10	0.4063	2.41	5.94	0.2265	1.91	8.45
sat	170.1563	508.21	2.99	382.3469	1234.62	3.23	84.8639	329.07	3.88
seg	10.2047	41.93	4.11	41.0937	88.45	2.15	7.4577	64.13	8.60
wav	23.103	147.23	6.37	68.2825	369.38	5.41	8.3548	83.94	10.05

TABLE VII
MOST ACCURATE NETWORKS AND THEIR SIZES: GAM

	MO-GGAM		GGAM		ssGAM	
	PCC	**Size**	**PCC**	**Size**	**PCC**	**Size**
1Ci/Sq	99.80	2	99.83	2	94.63	26
G4C-25	75.92	4	75.24	4	74.84	23
G6C-15	85.17	6	84.97	6	85.07	20
glass	76.00	8	73.44	9	68.75	14
Iris	94.90	2	94.85	2	95.21	7
page	96.38	5	96.34	6	94.52	7
pendigits	98.1	88	97.83	108	97.43	87
pima	82.67	2	76.72	2	72.41	3
sat	88.75	106	87.35	118	87.00	81
seg	92.59	13	92.71	17	91.29	31
wav	87.15	4	86.80	4	85.35	11

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 semi-supervised 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 I and II, 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 IV reveals). Finally, MO-GART is more elegant than GART because it does not require a user intervention to specify a-priori the preference towards one objective (accuracy) versus the other (size).

ACKNOWLEDGMENT

This work was supported in part by the NSF grants: 0341601, 0647018, 0717674, 0717680, 0647120, 0525429, 0203446.

REFERENCES

[1] S. Grossberg, "Adaptive pattern classification and universal recoding, ii: Feedback, expectation, olfaction, and illusions," *Biological Cybernetics*, pp. 187–202, 1976.

[2] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architectur for incremental supervised learning of analog multidimensional maps," *IEEE Transactions on Neural Networks*, vol. 3, pp. 698–713, 1992.

[3] G. Anagnostopoulos, "Novel approaches in adaptive resonance theory for machine learning," Ph.D. dissertation, University of Central Florida, Orlando, May 2001.

[4] J. R. Williamson, "Gaussian ARTMAP: a neural network for fast incremental learning of noisy multidimensional maps," *Neural Networks*, 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 Networks*, vol. 20, no. 8, p. 873891, October 2007.

[6] D. Goldberg, *Genetic Algorithms in search, optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[7] A. Kaylani, M. Georgiopoulos, M. Mollaghasemi, and G. Anagnostopoulos, "Genetic optimization of art neural network architectures," in *Proceeding of Artificial Intelligence and Soft Computing (ASC) 2007 Conference*, 2007, pp. 225–230.

[8] G. C. Anagnostopoulos, M. Bharadwaj, M. Georgiopoulos, S. J. Verzi, and G. L. Heileman, "Exemplar-based pattern recognition via semi-supervised learning," in *Proceedings of the 2003 International Joint Conference on Neural Networks (IJCNN '03)*, vol. 4, Portland, Oregon, USA, 2003, pp. 2782–2787.

[9] C. C. Coello, "Evolutionary multi-objective optimization: a historical view of the field," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 28–36, Feb. 2006.

[10] R. Santos, J. Nievola, and A. Freitas, "Extracting comprehensible rules from neural networks via genetic algorithms," *IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, pp. 130–139, 2000.

[11] 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 Optimization*, vol. 14, pp. 63–69, 1997.

[12] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*. Mahwah, NJ, USA: Lawrence Erlbaum Associates, Inc., 1985, pp. 93–100.

[13] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, p. 221248, 1994.

[14] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proceedings of the Fifth International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 1993, pp. 416–423.

[15] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A Niched Pareto Genetic Algorithm for multiobjective optimization," in *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, vol. 1.

[16] M. Erickson, A. Mayer, and J. Horn, "The niched pareto genetic algorithm 2 applied to the design of groundwater remediation systems," in *EMO '01: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*. London, UK: Springer-Verlag, 2001, pp. 681–695.

[17] H. Ishibuchi and T. Murata, "Multi-objective genetic local search algorithm," in *Proceedings of IEEE International Conference Evolutionary Computation, 1996*, May 1996, pp. 119–124.

[18] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A fast elitist Non-Dominated Sorting Genetic Algorithm for multi-objective optimization: NSGA-II," in *Proceedings of the Parallel Problem Solving from Nature VI Conference*.

[19] K. Deb, A. Pratab, S. Agrawal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions On Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr 2002.

[20] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[21] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," Gloriastrasse 35, CH-8092 Zurich, Switzerland, Tech. Rep. 103, 2001.

[22] J. E. Fieldsend, R. M. Everson, and S. Singh, "Using unconstrained elite archives for multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, p. 305323, june 2003.

[23] G. A. Carpenter and H. A. Tan, "Rule extraction: From neural architecture to symbolic representation," *Connection Science*, vol. 7, pp. 3–27, 1995.

[24] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz, "UCI repository of machine learning databases," 1998. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html