Enhancing the Diversity of Genetic Algorithm for Improved Feature Selection

Akram AlSukker, Rami N. Khushaba, and Ahmed Al-Ani
School of Electrical, Mechanical and Mechatronic Systems
University of Technology, Sydney (UTS)
Sydney, Australia
Alsukker, Rkhushab, Ahmed@eng.uts.edu.au

***Abstract*— Genetic algorithm (GA) is one of the most widely used population-based evolutionary search algorithms. One of the challenging optimization problems in which GA has been extensively applied is feature selection. It aims at finding an optimal small size subset of features from the original large feature set. It has been found that the main limitation of the traditional GA-based feature selection is that it tends to get trapped in local minima, a problem known as premature convergence. A number of implementations are presented in the literature to overcome this problem based on fitness scaling, genetic operator modification, boosting genetic population diversity, etc. This paper presents a new modified genetic algorithm based on enhanced population diversity, parents' selection and improved genetic operators. Practical results indicate the significance of the proposed GA variant in comparison to many other algorithms from the literature on different datasets.**

Keywords*— feature selection, genetic algorithm, premature convergan.

I. Introduction

Genetic algorithm (GA) is a powerful population based search procedure inspired by evolution theory. It proved to give good results when applied to different applications [1-6]. One of these applications is the feature/variable selection problem [2, 7-10]. In such a problem, a number of variables *d*, that perform the best under certain classification scheme, are selected from a pool of *D* variables ($d << D$). GA search consists of several steps: random population generation, fitness evaluation, fitness ranking, parent's selection, crossover and mutation operators, and a stopping criterion. A binary version of genetic population can be expressed as a list of binary strings with length *D* in which the presence of a feature is expressed by '1' and its absence is expressed by '0'. As an example, consider a binary string *X* with six features '110101', this represents selecting the first, second, fourth and sixth features.

In Simple Genetic Algorithm (SGA) search, the most fit population members have more chance to be selected to generate the next population through crossover and mutation operators. In certain complex applications, a high number of local minima are presented in different areas of the search space, which may get the genetic population stuck in one of them. In such a case, the newly generated population may not be able to produce better children, a situation known as "premature convergence", which is likely to happen when selecting a small number of features from a relatively large feature set. To clarify this problem, consider that GA is applied to a dataset of 10,000 features with a genetic population of 50 to search for the best 5 features. The number of features for a given generation will range between 5 and 250 depending on the duplication of features in the different population members. If the current population does not contain all of the five features that constitute the optimal subset, which is expected for this example, then it will be very likely for GA to get stuck in local minima, as the crossover operator does not introduce new features, and the possibility of finding an optimal feature using the mutation operator is very small. To overcome such a limitation a new method is proposed in this paper to enhance the diversity of the solutions represented by the population members.

This paper is organized as follows: section 2 presents variants of GA, section 3 describes the proposed GA. Experimental results are presented in section 4, and the conclusion is given in section 5.

II. Genetic Algorithm Variants

Studies in this field of research have shown that premature convergence has a great relation to population diversity. Simple GA search starts usually with diverse population but after a number of iterations, many population members tend to carry similar solutions and converge to a certain point. Some studies attempted to overcome this problem by one of the following techniques: (i) fitness scaling and parent selection, and (ii) improved crossover and mutation operators and introducing hybrid search. The Relative Fitness scaling Genetic Algorithm (GARF) introduced in [10] is inspired by the inverse power law relationship that occurs in physical and social science, as shown in equation 1

$$f(x) = \frac{x}{\sqrt{1 - \frac{x^2}{c^2}}}. \tag{1}$$

where x represents the fitness of population member and *c* is the ceiling factor which can take any values greater than or equal to the maximum possible fitness.

Note that when *c* is very high, the scaling factor's effect is minimum ($f(x) \approx x$). In this way relative scaling gives most individuals a small or average fitness and a very high fitness value to a few individuals. This scaling preserves population diversity by creating small differences between small and large fitness value members, and therefore avoids destroying unfit individuals early in the process that might contain good contributed features. Once candidate solutions approach a fitness value close to *c*, they will dominate the Roulette wheel.

A hybrid genetic search HGA method presented in [9] uses embedded local search to fine tune the search. HGA performs constrained crossover and mutation that produce children containing *n* features, where *n* is not too far from *d*, where d is the desired number of features. A local sequential search is then executed to repair the resulting outcome *X* by including exactly *d* features. Two local search operators are introduced called *ripple_add(r)* & *ripple_rem(r)*, where *r* is a constant number that reflects the number of features to be added/removed. In simple words, *ripple_add(r)* operator is implemented by adding *r* features one at a time to current set followed by removing *r-1* features, while *ripple_rem(r)* is utilized by removing *r* features followed by adding *r-1* features. The following procedure has been used by Seok et al. [9] to ensure the selection of fixed number of features (*d*) by each child.

- If *n*=*d*, apply *ripple_rem( )* then *ripple_add( )*.
- If *n* >*d*, apply *ripple_rem( ) n-d* times.
- If *n* <*d*, apply *ripple_add( ) d-n* times.

Where *n* represent the number of features presented in the generated child after crossover and mutation.

III. Proposed Genetic Algorithm

The proposed modified **D**iverse **G**enetic **A**lgorithm (**DGA**) tries to escape from local minima through introducing more diverse features to GA members and avoiding mating of exact members through a modified roulette wheel selection procedures. Additionally, a genetic population is checked when trapped in a certain area to get rid of duplicate population members and introduce a more diverse population; this procedure is summarized as follows:

- Apply simple genetic algorithm for a certain fraction of total number of iterations (threshold, denoted as *Th1*), to provide fast convergence to a solution, even if that solution presents a local minimum. In simple GA search the elitism will cause other population members to rapidly converge toward elite members. Therefore the elitism mechanism will be omitted in DGA after reaching this threshold and start to improve the search by trying to exit from local minima by partiality removing the pull toward best population members. *Th1* is set to 10% of the maximum number of iterations.
- When *Th1* is reached elitism will be eliminated from genetic search. Note that the fitness of the best member in the generated population might be lower than the previous iteration therefore a memory is introduced to save best achieved results. It is worth mentioning that this step is not required when applying SGA with elitism because elite kids perform as a memory.
- Modifying parent's selection algorithm in order to avoid selecting exact parent. Two roulette wheels are utilized. The first wheel selects the first parent *P1*. The second wheel that is used to select parent *P2* is constructed by eliminating the section that represents *P1*.
- Unconstrained crossover and mutation is used followed by repairing all resulting kids by adding features randomly from unused feature set, where each kid has different added features to enforce diversity.
- When the number of stall generations (No. of iterations without fitness improvement) exceeds a certain number, threshold *Th2*, the current population is checked in order to remove duplicate members and replace them with a random member generated from the unused features. A distance matrix is utilized for detecting duplicate members, i.e. when the Euclidean distance between two members are zero then one is replaced with a randomly generated set from unused features. *Th2* is set to 30% of the maximum number of iterations.

III. Experimental Results

A number of experiments have been implemented to investigate the performance of the proposed DGA against other GA variants. In the first experiment, a number of selected small UCI datasets were used (glass, vowel, wine), with a relatively low number of features. As a result a full search is feasible and can be done in less than a minute using a kNN classifier with 3 neighbors and take-one-out method as a fitness function, unlike other UCI dataset that requires a high computational cost to evaluate all subset, i.e. when selecting 3 features out of 16 from letter dataset around 3 hours is needed to evaluate all feasible subsets. Several GA variants have been compared according to the time required to achieve optimal solution found by Exhaustive Search (ES) as in Table 1. All GA variants were implemented using MATLAB, starting from the same initial population and allowing a maximum run time of one minute. A kNN classifier with 3 neighbors was used as a fitness function using take one out method. It is also worth mentioning that the search is repeated for five times and stopped when global minima is achieved as in full search. In all cases, GA variants were able to achieve global optima with a fraction of full search time. Table 1 show that ES achieved global optima faster than other methods when dealing with small search space, i.e. like selecting 2 or 8 features out of 10 in Glass and Vowel datasets as shown in Table 2. On the other hand, DGA was able to save up to 87% of the time, as when selecting 8 features from wine dataset to achieve the global minima. The justification behind the good performance of HGA is that better population members are achieved through local search, which helped in exploring most of the features. Nevertheless, the local search requires more subset evaluation when dealing with high dimension dataset as will be discussed in the next experiments.

The second experiment studies the performance of different GA variants when applied to a dataset which has a large number of redundant features. The Madelon dataset is employed for this purpose which contains only 5 relevant features out of 500 features. In Addition to GA variants, other benchmark search method such as plus l take away r (PTA), differential Evolution (DE) [11] and binary Particle Swarm Optimization (PSO) [12] have been evaluated. A brief description for each method is presented as follow:

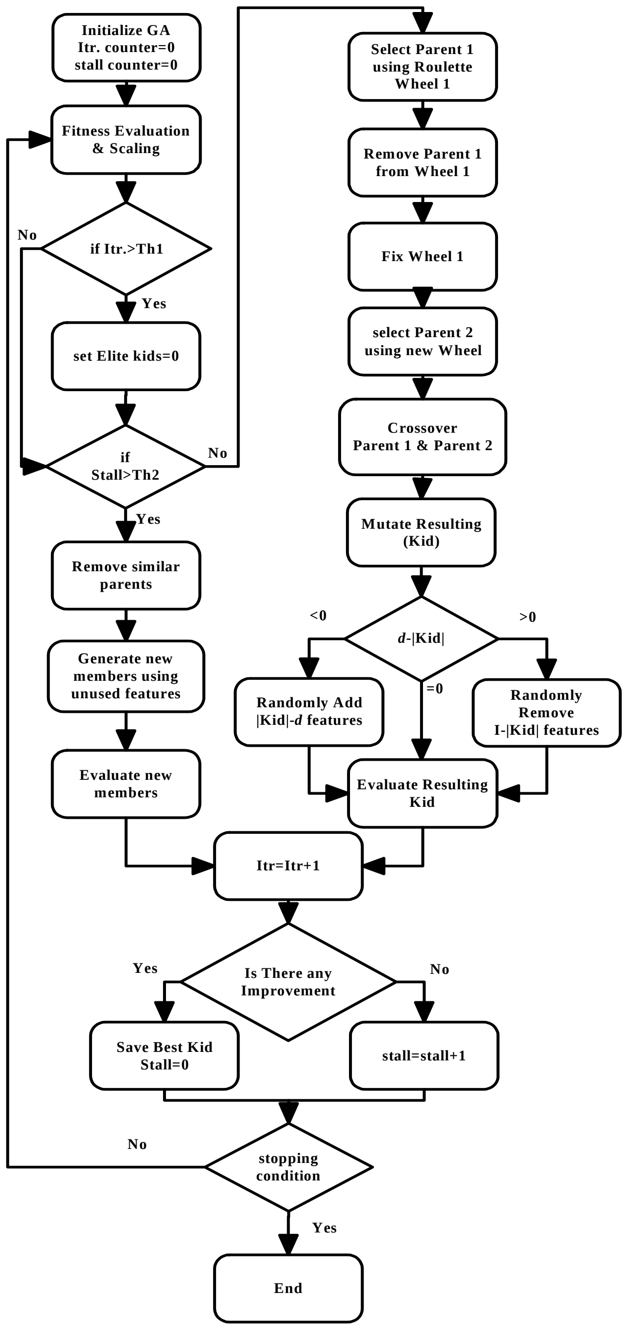

Figure 1. DGA flow chart.

-PTA: a refined sequential search approach that partially overcome the nesting effect and is based on applying a specific number of additions/removals of variables, where a feature that is not working well with other selected features can be removed. This step is repeated until the required number of features is achieved. This method is known as plus *l* take away *r* method (*l* > *r*).

TABLE I. CONVERGENCE SPEED OF GA VARIANTS

Dataset	*d*	ES Acc	Time (Sec.)				
			ES	SGA	HGA	DGA	GARF
Glass	2	99.53	1.5	3	**2**	**2**	4
D=10	4	99.53	6	3	3	**2**	3
	6	99.53	6	3	**2**	**2**	3
	8	99.53	1.5	3	**2**	3	7
Vowel	2	62.93	13	32	17	27	20
D=10	4	90.10	60	33	31	**23**	58
	6	96.57	62	54	**28**	32	29
	8	98.08	14	31	25	29	27
Wine	3	94.38	7	**2**	**2**	**2**	4
D=13	5	97.19	32	**6**	**6**	7	17
	8	95.51	30	5	**4**	**4**	7
	10	94.94	7	5	**2**	4	9

TABLE II. SEARCH SPACE SIZE & TIME SAVING

Dataset	N. of Patterns	*D*	*d*	No. of evaluated subset	$\frac{Time_{DGA}}{Time_{ES}}$
Glass	214	10	2	45	1.33
			4	210	0.33
			6	210	0.33
			8	45	2
Vowel	990	10	2	45	2.08
			4	210	0.38
			6	210	0.52
			8	45	2.07
Wine	187	13	3	286	0.29
			5	1287	0.22
			8	1287	0.13
			10	286	0.57

-DE: Differential Evolution (DE) is a simple, parallel, direct search, and easy to use optimization method having good convergence and fast implementation properties [11]. The crucial idea behind DE is a new scheme for generating trial parameter vectors by adding the weighted difference vector between two population members to a third member. The following equation shows how to combine three different, randomly chosen vectors (X_{r0}, X_{r1} and X_{r2}) to create a mutant vector $V_{i,g}$ from the current generation g:

$$V_{i,g} = X_{r0,g} + F \, . \, (X_{r1,g} - X_{r2,g}) \qquad (2)$$

Where F (0, 1) is a scale factor that controls the rate at which the population evolves. In addition, DE employs a uniform crossover, also known as discrete recombination, in order to build trial vectors out of parameter values that have been copied from two different vectors. In particular, DE crosses each vector with a mutant vector Y:

$$U_{j,i,g} = \begin{cases} V_{j,i,g} & \text{if } rand(0,1) \leq Cr \\ X_{j,i,g} & Otherwise \end{cases} \qquad (3)$$

The crossover probability $C_r \in [0,1]$ is a user defined value that controls the fraction of parameter values that are copied from the mutant. If the newly generated vector results in a lower objective function value (better fitness) than the predetermined population member, then the resulting vector replaces the vector with which it was compared

-PSO: A binary version of particle swarm optimization was adopted, the potential solutions (Particles) are considered as simple agents that can be presented by its location in N dimensional space. The goodness of each position is evaluated according certain measure (fitness function). Each particle memorizes its best achieved fitness (*Pbest*) and communicates with other particle to recognize the best achieved result (*Gbest*). Search started by randomly selection position, direction and speed of number of particles, each particle is accelerated toward *Pbest* (cognitive learning) and *Gbest* (social learning) while constantly checking the fitness of its current location. The probability of selecting a feature is calculated as according to next set of equations:

$$\begin{aligned} v_{ij} &= w * v_{ij} + c_1\, rand * (Pbest_{ij} - x_{ij}) \\ &+ c_2\, rand * (Gbest_{ij} - x_{ij}) \end{aligned} \tag{4}$$

$$P(v_{ij}) = \frac{1}{1+\exp(-v_{ij})} \tag{5}$$

$$if\ \ p_{ij} < P(v_{ij}(t))\ then\ x_{ij}(t) = 1;\ else\ x_{ij}(t) = 0 \tag{6}$$

where

- v_{ij} particle *i* velocity in the *j*th dimension.
- w is the inertia weight which is usually 1.
- c_1 and c_2 scaling factor which represent relative pull to *Pbest* and *Gbest*. c_1 and c_2 usually set to 2.
- *rand* () random variable between 0 and 1.
- X_{ij} particle *i* position in the *j*th dimension.
- $Pbest_{ij}$ and $Gbest_{ij}$ represent the personal and global best in the *j*th dimension.
- p_{ij} is a vector of random numbers, drawn from a uniform distribution between 0.0 and 1.0

The graphs shown in Figure 2 indicate that DE and DGA converge faster than other methods. However, DGA achieved its maximum using around 100 iterations while DE required an additional 50 iteration to achieve the same result. Figure 2 also shows that DE and DGA achieved the best accuracy (90.6%) followed by PTA, GARF and HGA (89.5%) and the worst result were achieved by SGA (88.23%). In one hand, HGA did not perform well in this experiment and takes around 200 iteration for quick convergence due to the negative aspect of local search that search all surrounding search space that only provides a small improvement increment. This may indicate that HGA has a limitation when applied to large feature sets with high redundancy taking into account that local search will produces large number of subsets that need to be evaluated each crossover and mutation operation. The number of subsets to be evaluated (*NoS*) increases as number of features increases and as ripple factor *r* increases. Equations 7-10 illustrate the complication associated with increasing both ripple factor and feature space.

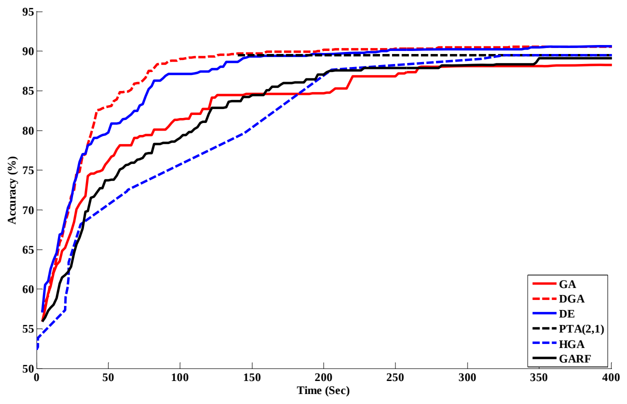

Figure 1. Convergence of different methods when selecting 5 features from Madelon.

$$NoS(ripple\_add(r)) = \sum_{i=0}^{r-1} D - (|X| - i) + \sum_{i=0}^{r-2} (|X| + r) - i \tag{7}$$

$$NoS(ripple\_rem(r)) = \sum_{i=0}^{r-1} |X| - i + \sum_{i=0}^{r-2} D - (|X| + r) - i \tag{8}$$

Note that for *D*>>|X|

$$NoS(ripple\_add(r)) = r \times D \tag{9}$$

$$NoS(ripple\_rem(r)) = (r-1) \times D \tag{10}$$

On the other hand, GARF will slow down the convergence of search due to the fact that better population members are assigned lower fitness ranks to encourage them evolve with less fit members. Also, GARF does not take into consideration the size of search space, and hence some features will not have the chance to be explored.

In the third experiment, a number of large datasets were used to study searching huge search spaces. Eleven gene datasets obtained form (***http://www.gems-system.org/***) were used. The number of features varies in those dataset from 2 to 12 thousands with relatively small number of test pattern (60-308 patterns) as shown in Table 2. Due to the small number of patterns in these datasets a *10-fold* cross-validation approach with kNN classifier (*k=3*) was adopted. The results in figures 3-13 represent the average value of ten runs, with all methods starting from the same initial population to make a fair comparison. The search was started by selecting 5 to 50 features with 5 steps increment which only represent less than 2.5% of the total number of features.

HGA has been excluded from this experiment due to its high computational cost associated with large number of subset evaluation after each kid generation especially when dealing with high dimension dataset coupled with expensive fitness function as explained earlier.

TABLE III. GENE DATASET DESCRIPTION

Dataset	Symbol	NoP	NoF	NoC
Lung Cancer	LC	203	12600	5
Prostate Tumor	PT	102	10509	2
Leukemia 1	L1	72	5328	3
Leukemia 2	L2	72	11226	3
Brain Tumor 1	BT1	90	5920	5
Brain Tumor 2	BT2	50	10368	4
9 Tumor	T9	60	5726	9
11 Tumor	T11	174	12533	11
14 Tumor	T14	308	15009	26
DLBCL	DL	77	5469	2
SRBCT	SR	83	2309	4

In order to analyze the results one can start by categorizing the performance of the different feature selection methods into three categories. In the first category, it can be noticed that all of the SGA, GARF, and BPSO compete with each other on average across all of these datasets, while at the same time providing the lowest performance in comparison to the other utilized methods. The second category includes the proposed DGA and DE, where those two methods are again showing competing performance across the different datasets, while at the same time providing better performance than the methods included in the first category. The third and the final category is occupied by the PTA(2,1) method, with practical results indicating that PTA(2,1) offered absolutely the best classification performance. However, it is worth mentioning here that PTA(2,1) is a very time consuming method, i.e. PTA(2,1) results on T11 were achieved in 1.62 hours, while SGA, GARF, DGA, PSO and DE took 12.2, 11.6, 13.6, 19.8 and 13.6 minutes respectively which is around %15 of the PTA search time. In most cases, DGA outperformed the other GA variants when selecting a range of small feature sets (5-50) out of thousands of features. DGA was also able to achieve results close to DE, which claims to outperform GA in many studies.

IV. CONCLUSION AND FUTURE WORK

We proposed in this paper a modified genetic algorithm method for feature selection; terms as DGA, which aimed at overcome the premature convergence problem that traditional GA suffers from. It basically infuses more diverse population members when needed. Results showed that DGA was able to outperform other GA variants with similar computational cost and achieved results close to that of differential evolution (DE). Other GA based methods such as DGA and GARF suffer from limitation when applied to high redundant data with very high dimensionality, i.e. high computational local search made DGA search impractical and the fitness scaling in GARF compromise the convergence of the search by allocating lower fitness rank to the best population members. DGA algorithm will be further developed in the future to enhance the parent selection procedure and to fuse better feature when needed.

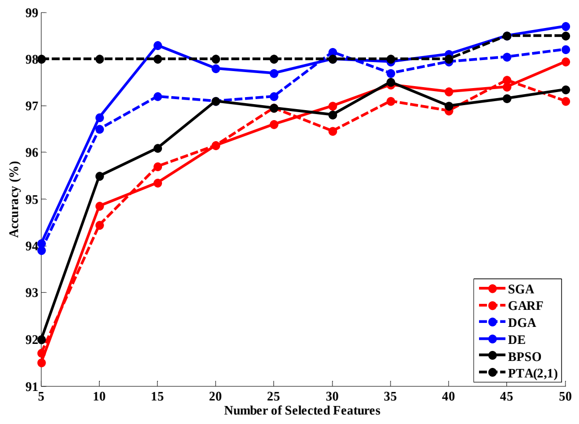

Figure 2. Classification performance of Lung Cancer dataset.

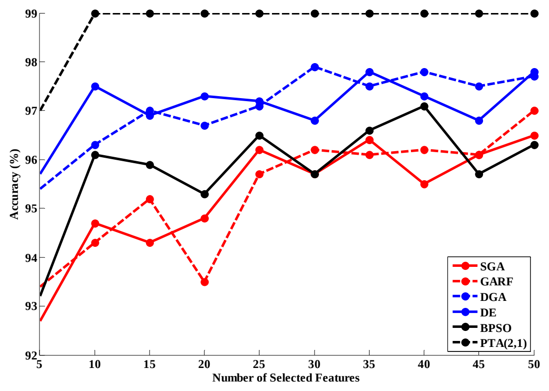

Figure 3. Classification performance of Prostate Tumor dataset.

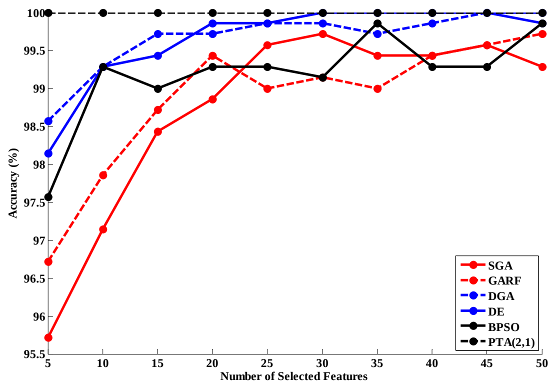

Figure 4. Classification performance of Leukemia 1 dataset.

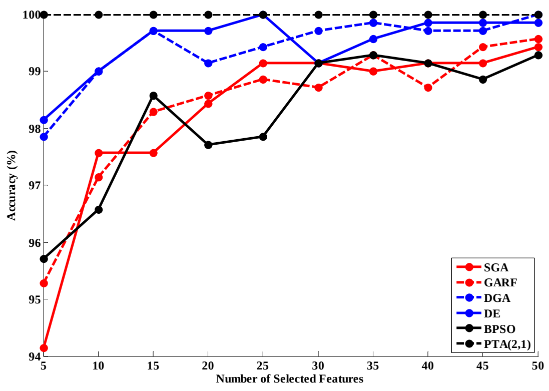

Figure 5.   Classification performance of Leukemia 2 dataset.

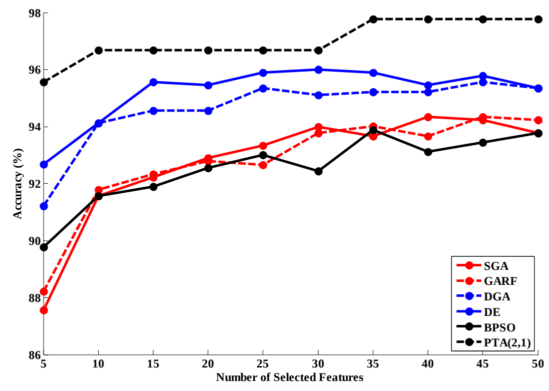

Figure 6.   Classification performance of Brain Tumor 1 dataset.

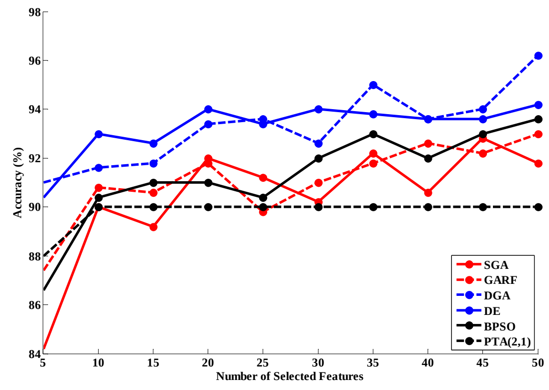

Figure 7.   Classification performance of Brain Tumor 2 dataset.

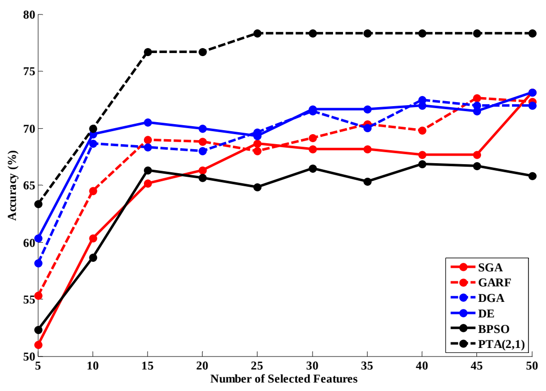

Figure 8.   Classification performance of 9_Tumor dataset.

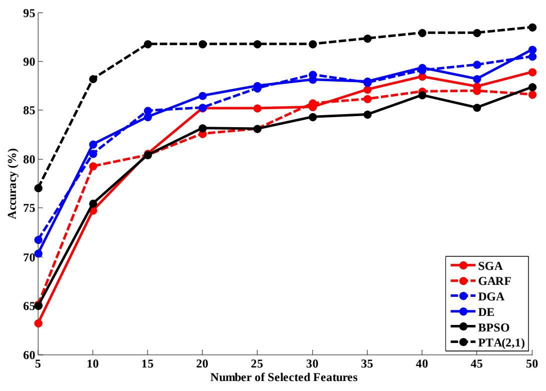

Figure 9.   Classification performance of 11_Tumor dataset.

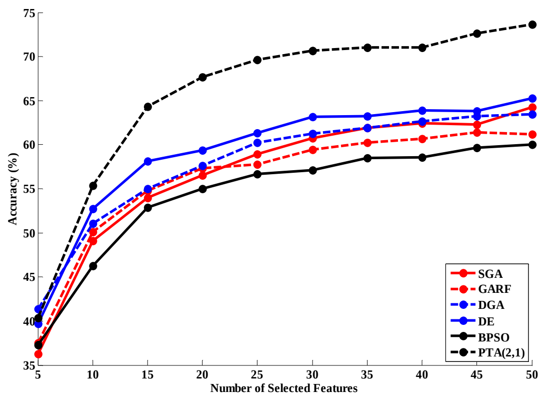

Figure 10. Classification performance of 14_Tumor dataset.

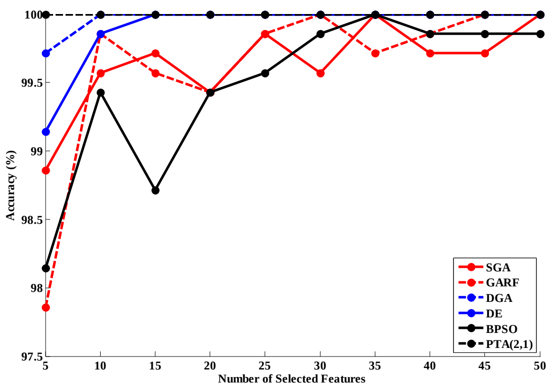

Figure 11. Classification performance of DLBCL dataset.

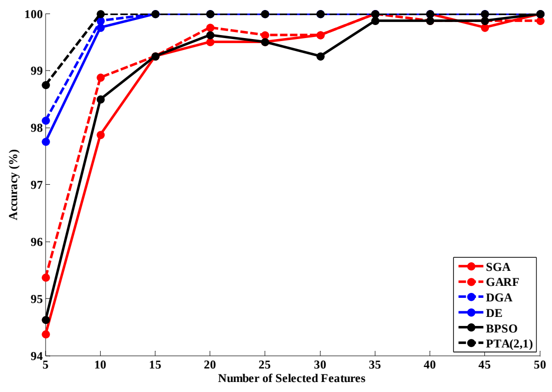

Figure 12. Classification performance of SRBCL dataset.

References

[1] A. S. Andreou, E. F. Georgopoulos, and S. D. Likothanassis, "Exchange-Rates Forecasting: A Hybrid Algorithm Based on Genetically Optimized Adaptive Neural Networks," Computational Economics, vol. 20, pp. 191-210, 2002.

[2] M. T. Harandi, M. N. Ahmadabadi, B. N. Araabi, and C. Lucas, "Feature selection using genetic algorithm and it's application to face recognition," in Cybernetics and Intelligent Systems, 2004 IEEE Conference on, 2004, pp. 1368-1373.

[3] C. G. Wu, X. L. Xing, H. P. Lee, C. G. Zhou, and Y. C. Liang, "Genetic algorithm application on the job shop scheduling problem," in Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on, 2004, pp. 2102-2106 vol.4.

[4] W. Li-Ying, Z. Jie, and L. Hua, "An Improved Genetic Algorithm for TSP," in Machine Learning and Cybernetics, 2007 International Conference on, 2007, pp. 925-928.

[5] S. Obayashi, D. Sasaki, Y. Takeguchi, and N. Hirose, "Multiobjective evolutionary computation for supersonic wing-shape optimization," Evolutionary Computation, IEEE Transactions on, vol. 4, pp. 182-187, 2000.

[6] B. T. Skinner, B. T. Skinner, H. T. Nguyen, and D. K. Liu, "Classification of EEG Signals Using a Genetic-Based Machine Learning Classifier," in Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, 2007, pp. 3120-3123.

[7] A. AlSukker and A. Al-Ani, "Evaluation of Feature Selection Methods for Improved EEG Classification," in Biomedical and Pharmaceutical Engineering, 2006. ICBPE 2006. International Conference on, 2006, pp. 146-151.

[8] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," Intelligent Systems and their Applications, IEEE, vol. 13, pp. 44-49, 1998.

[9] O. Il-Seok, L. Jin-Seon, and M. Byung-Ro, "Hybrid genetic algorithms for feature selection," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 26, pp. 1424-1437, 2004.

[10] G. Surabhi, "Relative fitness scaling for improving efficiency of proportionate selection in genetic algorithms," in Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers Montreal, Québec, Canada: ACM, 2009, pp. 2741-2744.

[11] R. N. Khushaba, A. Al-Ani, and A. Al-Jumaily, "Differential evolution based feature subset selection," in Pattern Recognition, 2008. ICPR 2008. 19th International Conference on, 2008, pp. 1-4.

[12] B. Coppin, Artificial intelligence illuminated, 1st ed. Boston: Jones and Bartlett Publishers, 2004.