Reduction Technique for Instance-based Learning Using Distributed Genetic Algorithms

Tahseen A. Al-Ramadin

Al-Hussein Bin Talal University, Ma'an 71111, Jordan
tahsen@ahu.edu.jo, tahsenr@yahoo.com

Abstract

This work addresses the problem of instance reduction using a distributed implementation of genetic algorithms. Different existing parallel and distributed models for parallelizing genetic algorithms are investigated and applied here to solve the problem of instance reduction. a new parallel model is proposed and implemented we called Global Control Parallel Genetic Algorithm. The results showed enormous reduction in time of 90% over the other models. The resulted dataset showed an acceptable accuracy results on average over all datasets. The model achieved a better reduction in dataset size of 90.22% compared to the other models that didn't get better that 87.91%. Thus, the proposed distributed system model for instance reduction showed better performance over all model in reducing the time and even reducing the training dataset size while maintaining the same level of accuracy of the original sequential genetic algorithm.

Keywords: *Instance-based; Parallel Genetic Algorithm; instance reduction; training set.*

1. Introduction

Instance-based learning is one form of machine learning that uses past observations to classify new observations. Instance-based learning algorithms are trained on a set of data called training set which consists of several instances each of which consists of a group of attribute values and a class value. The duty of the algorithm is to predict the class of a new instance by comparing its attribute values with those of previous learned instances [1] [2]. In most learning problems, the number of instances in the training set is large, beside this, the number of output classes can be noisy or inaccurate, can be redundant, can be of different types, and can have different valuations. Thus, the quality and quantity of training data have direct impact on the learning process [3].

Instance-based learning (IBL) algorithms have the advantage of being able to rapidly include the most up-to-date instance and yield relatively accurate predictions. However, they suffer from poor scalability for large number of instances in the dataset. Despite that IBL algorithms give good classification accuracy, they suffer the following deficiencies: Requiring large storage requirements since all instances are stored, the amount of processing they require for classification, and reduced classification accuracy since noisy instances are also stored.

Because of these shortcomings of IBL algorithms, instance reduction algorithms are used to improve the efficiency and accuracy of the learning system [4]. The instance reduction algorithm attempts to discard irrelevant and noisy instances which do not harm the classification accuracy. Thus, the result is smaller training set that requires less storage and less processing time.

One of the first attempts to reduce the training set size is the Condensed Nearest Neighbor (CNN) proposed by Hart [9]. The main goal of this method is to obtain the reduced and consistent set of prototypes to be used with the 1-NN rule without error in the training set or with the K-NN rule without significantly degrading its performance. The whole process is iterated until there is no change in a complete pass through the initial training set. The method does not guarantee a minimal consistent subset. In addition, the final size as well as the composition of the final condensed subset depends on the order of data presentation [9]. An improvement of the CNN was introduced by Ritter and Tomek [10], called Selective Nearest Neighbor (SNN). The improvement is achieved by growing the condensed set using only the patterns that are close to the decision boundary. However, the algorithm still suffers from its sensitivity to noisy instances and it is more complex than most other algorithms. [1][2][3].Moreover, many instance reduction techniques have been suggested to address the problem of noise. Wilson and Martinez [4], proposed six reduction algorithms, DROP1-5 and DEL, used to reduce the size of the training set while maintaining or even improving classification accuracy.

2. Genetic Algorithms

Genetic Algorithms (GAs) are the most popular evolutionary algorithms and have been widely used to solve difficult optimization problems. Moreover, they are search methods that are based on the principles of the natural selection and genetics [4]. Genetic algorithms encode the decision variables of the search problem into finite-length strings of alphabets of certain cardinality . The strings, which are the candidate solutions to the search problem, are referred to as chromosomes, and the alphabets (digit) are referred to as genes . To evolve good solutions and to implement natural selection, we need a measure for distinguishing good solutions from bad solutions. The measure could have an objective function that is a mathematical model or a computer simulation, or it could have a subjective function as humans choose better solutions over worse ones [16]. Fitness measure determines the candidate solution’s relative fitness, which will subsequently be used by the GA to guide the evolution of good solutions [8]. Once the problem is encoded in a chromosomal manner and using a fitness measure, good solutions can be selected and distinguished from bad solutions.

3. Parallel Genetic Algorithms

There are three reasons for investigating Parallel Genetic Algorithms PGAs. Firstly, to reduce the execution time of the calculations associated with GA in parallel [6]. Secondly, to maintain diversity and reduce the probability of the premature convergence. This can be done by utilizing a number of independent subpopulations and by using local selection rules, which allow a chromosome to mate with chromosomes in its local neighbourhood. Thirdly, to mimic the natural evolution more closely. In nature, a population is typically many independent subpopulations that occasionally interact. There are several ways to parallelize GAs. The simple way is to parallelize the main operations of GA (i.e., evaluation of fitness values, reproduction, crossover, and mutation) that creates the next generation from the previous one. Parallelizing the computation of fitness values and performing crossover and mutation can be easily done. In the reproduction step, to select parent chromosomes all chromosomes in the population have to be accessed by all processing elements, this can be a parallel communication bottleneck. Therefore, this is only suitable for tightly coupled fine grain parallel machines.

An advantage of PGAs over sequential genetic algorithms is the ability for solutions to be migrated amongst GAs in the system. This is particularly useful in the search, which may be fixed, in a local optimum leading to a poor solution. Migration of candidate solutions increases the probability of breaking out of a local optimum among slaves. Thus, migration of genetic material gives many kinds of new children in the population .

The basic idea behind the parallel genetic algorithm is to divide the GA task into subtasks and use different processors to execute each subtask [11]. A parallel GA is an algorithm that has multiple component GAs, regardless of their population structure, and a component GA is usually a traditional GA with a single population. Its algorithm is augmented with an additional phase of communication code so as to be able to convey its result and receive results from the other components .

In general, there are three approaches to implement a GA on a parallel machine :

- Single-population and parallelizing fitness evaluation. In this approach, one population is kept in one node and evaluating the fitness function is distributed over the other nodes.
- Multiple-population: divides the global population into several sub-populations and evolves a sub population on each processor .
- Global population while parallelizing the GA operators: Those operators are restricted on the neighboring individuals. This method is categorized as cellular evolutionary algorithms that are the extension of Multiple-population approach.

4. Parallel Genetic Algorithm Models

Parallel genetic algorithms found in the literature can be classified according to the distribution of population among the processing elements and the way the GA operators are applied. The population can be divided into several subpopulations, or there is one population, but each population member interacts only with a limited set. In some implementations, migrant chromosomes may move only to geographically nearby subpopulations, rather than to any arbitrary subpopulation . Another way of distribution is according to the GA operators as each node may perform all the operators. Parallel genetic algorithms are classified into three categories or models: Island, Migration, and Neighborhood [7].

In general, all genetic algorithms perform the same set of steps to solve certain problem. The differences are the detailed implementations of each step. The genetic algorithm steps specific to the problem of instance reduction are as follow:

1. **Representation and Initial Population**: The initial population is chosen randomly after deciding the population size. Generally, the dataset in any instance-based learning system is huge. Therefore, the population size must be enough to give a good result within acceptable time. In our implementation, the population size will be dynamic and depends on the dataset size and the training set size. Other variable is the chromosome size (no of genes in the chromosome). The chromosome size is fixed and not changed in all the algorithm operations. Chromosome consists of genes (one for each instance in the dataset) with two possible states: 0 and 1. If the gene is 1, the its associated instance is included in the training set. If it is 0, then the associated instance is not included

2. **Fitness Function**: This used to evaluate each chromosome in the population. The specification of the function depends on the problem domain. In our case the evaluation is to see whether the candidate training set is suitably represents the original dataset. Consequently, the fitness function performs the following operations:

   Step 1: Create a copy from the original dataset as test dataset.

   Step 2: Determine the classes of the test dataset using K-NN.

   Step 3: Compare the estimated classes with the actual classes (reduction ratio and accuracy)

   Step 4: Compute the fitness function value.

3. **Crossover** operations are used to create new offspring (children). This is done by combining a portion of two parent's chromosomes. Crossover input is a pair of randomly selected parent chromosomes and the output is a pair of offspring that combine the parents' features. Crossover operation consists of the following steps:

   Step 1: Determine the crossover probability.

   Step 2: Select pairs of parent chromosomes randomly from the population.

   Step 3: Generate a crossover point(s) randomly.

   Step 4: Generate the first offspring.

   Step 5: Generate the second offspring.

4. **Mutation** introduces small changes in chromosomes. One of the populations is randomly selected and mutation is applied to it with low probability. Mutation operation has several steps:

   Step 1: Determine the mutation probability.

   Step 2: Randomly choose chromosomes from current population.

   Step 3: Randomly choose a gene in the selected chromosome.

   Step 4: Mutate the selected gene.

5. **Selection**: In order to decide which population members to be moved to the next generation, a method of selection has to be chosen. Selection allocates more reproduction of those solutions with higher fitness values and thus imposes the survival-of-the-fittest mechanism on the candidate solutions. The main idea of selection is to prefer better solutions to worse ones. Many selection procedures have been proposed to accomplish this, including roulette-wheel selection, stochastic universal selection, ranking selection, and tournament selection.

In our work, the main job of the selection is to keep the population diverse enough to be able to move out of local minima. A mixed method (close to Elitism) is used which selects the first chromosome with the best fitness value then the other chromosomes will be chosen randomly using the roulette method.

The selection operator has five steps as follows:

Step 1: Compute the sum of fitness functions.

Step 2: Move the chromosome with highest value to the next generation.

Step 3: Generate a random number.

Step 4: Move the chromosome corresponding to the slot in the wheel to next generation.

Step 5: Repeat Step 4 until the population size is reached.

The algorithm considers all the instances in the dataset and the subset instances are comprised according to the chromosome. Each instance in the original dataset will be checked against the subset to find the proposed classes. By comparing the original classes with the projected ones, if it is the same then this subset can represent this instance. Dividing the number of correctly classified instances by the dataset size gives the accuracy of the training dataset. The following algorithm is used to compute the accuracy for the subset represented by a chromosome:

Step 1: Get a chromosome from the population poll.

Step 2: Scan the genes in the chromosome from left to right, if the gene equals to one then copy the corresponding instance from the dataset to the subset proposed by that chromosome.

Step 3: Repeat Step2 until the end of the chromosome.

Step 4: For each instance in the original dataset predict the class of that instance using the K-NN algorithm on the subset.

Step 5: Count the number of correctly predicted classes in Step4.

Step 6: Compute the accuracy by dividing the number of correctly classified instances by the number of instances in the original dataset.

In our application we have many constrains such as the population size, crossover probability, mutation probability, and the number of generations (method for convergence). A dynamic (random) population size, crossover probability, and convergence method are used. The mutation probability is also dynamic but it will not be more than 0.1.

The convergence in our work is a very important factor. Because we want to get the best representative training set for the original dataset with minimum time and the least possible storage. The datasets are very large which have many parameters (i.e., attributes, classes, and the dataset size). Therefore, we must be careful in choosing the terminating criteria. Convergence criteria with many parameters will be used then compared with each others. The following criteria will be used to stop the genetic algorithm:

- Number of generations: The algorithm stops when the number of generations reaches a certain value, for example, 100, 1000, and 10000 generations will be tested and then compared.
- Freeze generations: The algorithm stops if there is no improvement in the objective function for a sequence of generations. In this case the algorithm will be given a percent value (80%, 85%, 90%, 95%, and 98%) and the number of generations.
- Freeze time limit: If there is no improvement in the objective function during ten generations the algorithm will be stopped.

5. Parallelizing Instance Reduction

Problems with huge number of instances take very long running time to produce a representative dataset of small size. In this section, a new technique will be introduces to get the result with a reasonable short time by employing several parallel genetic algorithm models. These models will be implemented and applied on instance reduction problems. For each model, it will be shown how the original dataset is distributed among the processing elements of the distributed system. We will also show how the genetic operations are performed by each processing element and how the results are collected.

Island Model: In this model, each node will perform the operations of the sequential instance-reduction genetic algorithm. This model can be implemented using two variations of distributing the original dataset. The first method is to distribute a copy of the whole dataset for each processing node. The second method is to distribute disjoint subsets of the original dataset among the processing nodes. In the first method, a copy of the original dataset is saved in each processing node. The final results are collected from each node then choose the best result among them. This model is shown in figure 1.

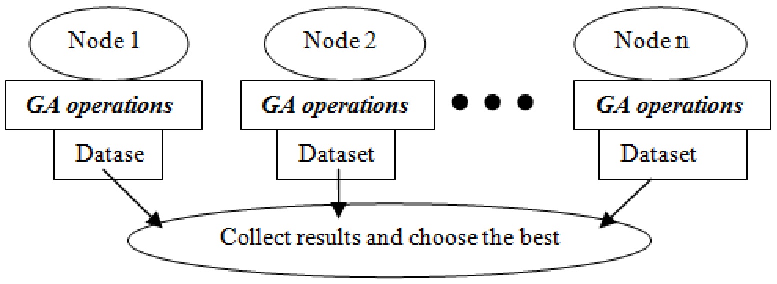

Figure 1 The Island model with a copy of the original dataset

The results produced by applying this model are somehow closed to the sequential genetic algorithm. In some cases of large dataset size we got a slight improvement in time reduction compared to pure sequential algorithm. In the other hand, for small datasets, the sequential algorithm shows better results. These results are due to network transmission delay incurred during the initial and final phases of distributing the dataset and collecting the results from processing nodes.

A different implementation method of this model has been performed, in which disjoint subsets of the original dataset are distributed among the processing nodes. The choice of division and distributing the subsets can be done in one of the following techniques:

1. Each node gets an equal-size subset from the original dataset. The number of subsets depends on the number of the nodes in the system.
2. Each node has a subset of the dataset in which instances are chosen randomly.
3. Each node has all the instances of one particular class. The number of subsets will depend on the number of different classes in the dataset.

In all of these choices, the union of all subsets will be equal the original dataset. The Island model implementation using the second technique is shown in Figure 2.

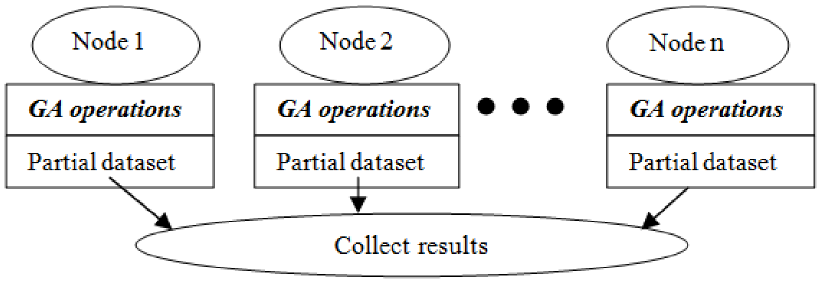

Figure 2 The Island Model with Partial Dataset

The division of the dataset is done by a separate processing node then each subset is transferred to its corresponding node. When all processing nodes are done with the application of the GA, the results are collected from each node and then combined by a union operation to get the final result.

Neighborhood model: In this model, the original dataset are distributed on a number of nodes. After initializing the first generation of the genetic algorithm, each node sends a selection (best) of chromosome(s) to its next neighborhood node. The neighboring node in turn adds the chromosome to its population. If this received chromosome (the new one) is better than the one(s) it has, then that chromosome might have a good chance to live. This synchronization between the subsets can improve the chances to get better results. At the end of the genetic algorithm execution, the best results of all nodes are considered the model result. Figure 3 describes the Neighborhood model:

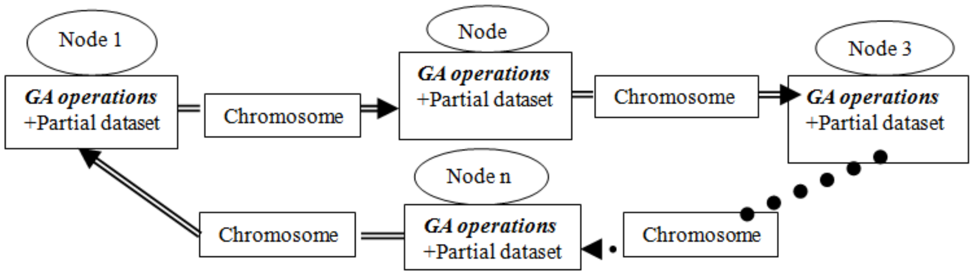

Figure 3 The Neighborhood Model

Notice that the data transfer between the nodes would give an improvement in the results since the best chromosomes are being exchanged by adjacent nodes in the system giving them a chance to produce an enhanced new generations. Hence, improving individual node's result leads to improving the overall result of the model.

Migration Model: Migration model allows each node to interchange the chromosomes with a range of neighboring nods. This relaxation over the Neighborhood model led to a more improvement in the results. In general, the best chromosomes are transferred between nodes. When each node has all the instances of a dataset, then all nodes can share the best chromosomes resulted at each node so far. In the case that each node has partial dataset then chromosomes can be shared between semi-isolated groups having the same part of data. Each processing node contains a Migration algorithm to be able to exchange data with other nodes.

Each node has a list of addresses of all other nodes from which it can randomly select a node for exchanging chromosomes. Figure 4 shows the Migration model.

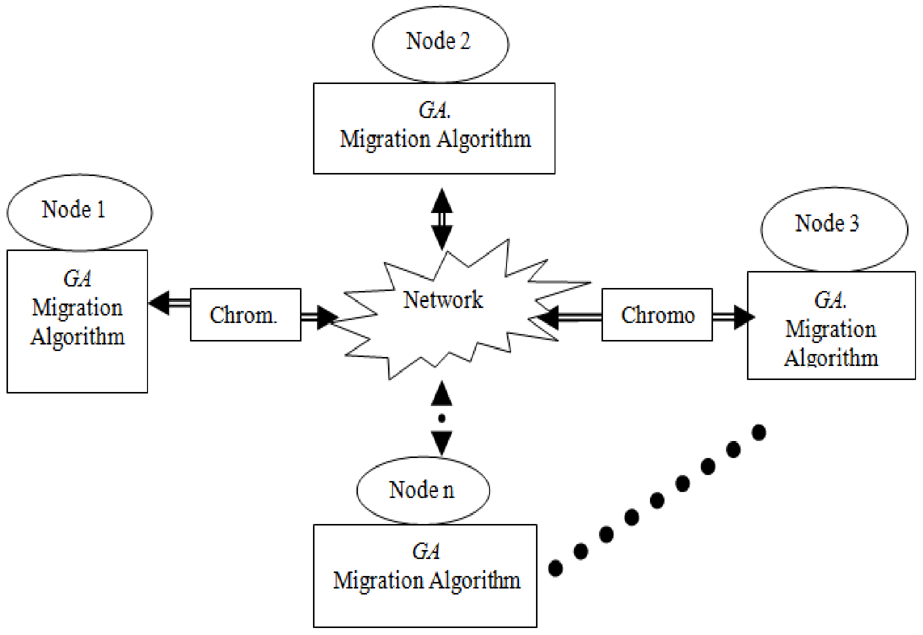

Figure 4 The Migration Model

Global-Control Parallel Instance-Reduction

All the above models do not have global control of operations of the processing nodes. Each node performs the GA independent of the others except that from time to time nodes exchange part of the population. In this section we describe a new parallel genetic algorithm model that can be used for solving the problem of instance reduction and possibly any genetic algorithm problem.

Global Control PGA has a manager node, the *control-node* that controls all the other nodes in the system. The control-node has three parts; *distribute*, *control*, and *collect*, described below:

1) *Distribute*: Distribute the dataset to all other nodes according to a distribution criterion. This is performed on the initial population and before producing the first generation. The distribution criteria are:

   1.1 The control node sends a copy of the whole dataset to each node in the system. Thus, each node starts with all instances of the dataset.

   1.2 The control-node divides the dataset into equal parts (the selection from the dataset may be sequential or random) and distributes them among the nodes in the system. Thus, each node starts with part of the dataset.

   1.3 The control-node divides the instances in the dataset according to the classification type of the instances. Depending on the total number of nodes and the number of different classification types in the dataset, the control-node may send the instances of one or more classification types to some nodes in the system. Thus, each node starts with one classification type.

2) *Control*: Exchanging the chromosomes with the other nodes. All other nodes will run a sequential genetic algorithm working on its own part of the dataset. The control-node gets part of the chromosomes (generally the best) from the first node and keeps it as the current global solution. Contacting with the second node and get the best solution it has. Compares this solution with the current global solution it already has, and decides whether to accept the new solution. If the new solution is better than the current global solution then it considers it as the new global solution. Otherwise, it sends the current global solution to that node with the corresponding chromosomes. This process continues until the control node decides to stop when the termination condition is satisfied.

3) *Collect*: Finding the final result of the model. The termination of the algorithm depends on the termination criterion used. The control-node performs the following algorithm for controlling the system and exchanging the data between the nodes.

```
Step1: Distribute the dataset amongst the system nodes.
Step 2: Get chromosome from node-$i$, consider it as global solution.
Step 3: Listening to all nodes until node-$j$ sends notification.
    Step 3.1: Connect to node- $j$ and get the solution.
    Step 3.2: Compare global solution with node-$j$ solution
    Step 3.3: If node-$j$ solution is better than global solution then set node-$j$'s
              solution to be the global solution. Otherwise, send current global
              solution to node-$j$.
Step 4: Repeat step3 until termination criterion is satisfied.
Step 5: If termination criterion is satisfied then send stop command to all nodes.
```

The values of *i* and *j* are the node number in the distributed system that can be between 1 and the total number of nodes in the system, *n*.

Each node *l*, in the system, performs the following algorithm steps:

```
Step1: Receive dataset from the control-node.
Step2: Perform the sequential GA algorithm.
Step3: Notify the control-node at the end of the generation.
Step 4: Exchange the chromosomes with control-node when requested.
Step5: If new chromosome arrived from control-node (global solution) then accept
    it as a local solution.
Step 6: Repeat Steps 2 to 5 until control-node sends stop command.
```

Figure 5 shows the Global Control PGA model. The Figure shows the operations of the control-node and its relation with the other nodes.

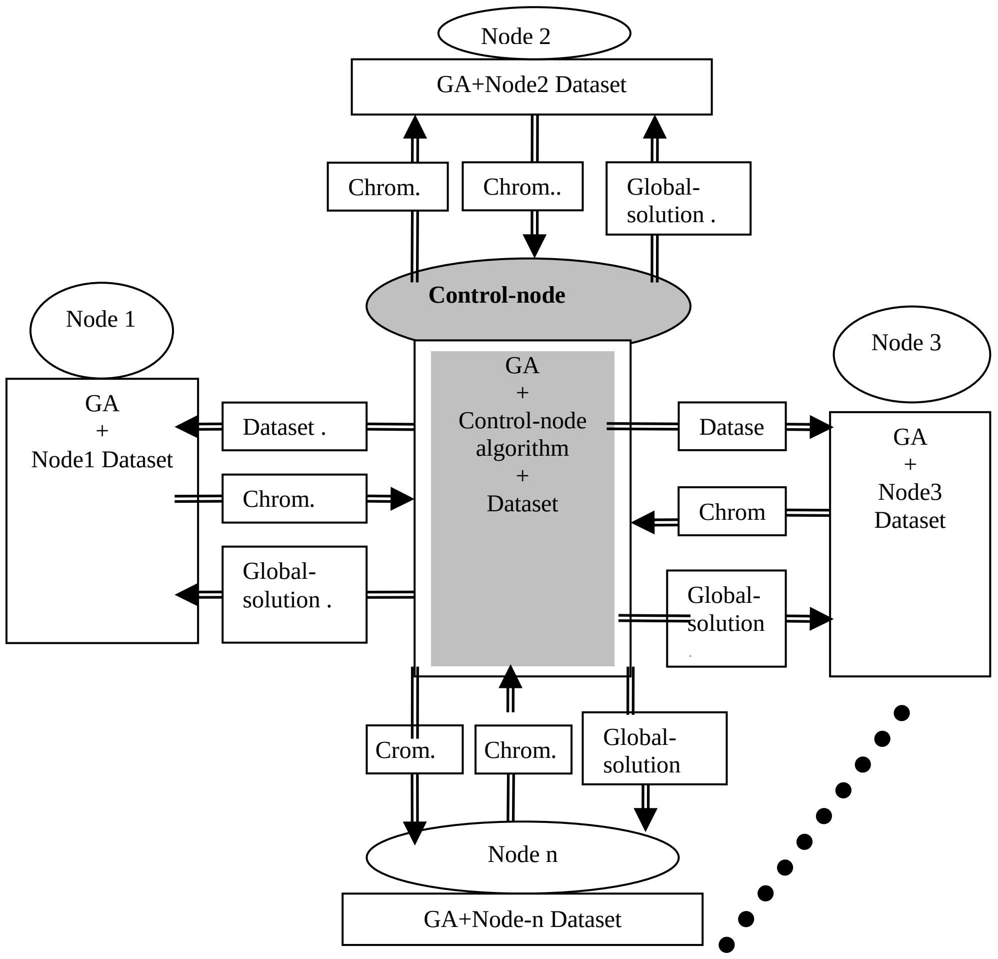

Figure 5: The Global Control PGA

The control node may use various methods to distribute the dataset (i.e., the first populations) among the nodes. These methods are:

-Each node starts with all the dataset. In this method. The control node sends a copy of the whole dataset to each node in the system.

-Each node starts with part of the dataset. In this case, the control node divides the dataset into equal parts (the selection from the dataset may be sequential or random) and distributes them among the nodes in the system.

-Each node starts with one classification type. That is, the control node divides the instances in the dataset according to the classification type of the instances. Then, it may send the instances of one or more classification types to each node in the system depending on the total number of nodes and the number of different classification types in the dataset.

6. The Experiments and Performance Measures

The reduction algorithms in all experiments were tested on 25 datasets of various sizes from the Machine Learning Database Repository at the University of California Irvine[17].

The statistics of these datasets are shown in Appendix A. The tests were performed on a distributed system based on a LAN of Personal Computers running MS Windows operating system. The number of stations varies according to the model, distribution policy, and the dataset size. The algorithms were written in Java programming language that supports our distributed system using the RMI feature. The results collected were the average of 100 runs/trials for the small datasets (less than 1000 instances) or the average of 10 runs for the other datasets.

Performance measures used in testing the systems were computed. These measures are: Classification accuracy which is the percentage of the instances that are classified correctly. The accuracy is calculated by dividing the Number of correctly classified instance using the training dataset by the Number of instances in the original dataset the multiply the result by 100%. The time was the execution time of the whole algorithm measured in hours, minutes, seconds, and milliseconds. The reduction in time which is the percentage of reduction that is obtained when using, in general, mothod2 instead of method1 and computed using the following formula:

(Time of method1 – Time of method2)/ Time of method1 * 100%.

Finally, size reduction is the percentage of reduction in size obtained using a certain method instead of another method and computed as follows:

Size reduction using method2 = (Size obtained by method1 – Size obtained by method2) / (Size obtained by method1) * 100%.

When comparing the results of the proposed model with the other models, the results of the Global Control PGA showed good classification accuracy, almost similar to the sequential algorithm. This good classification accuracy was expected, since the basic reduction algorithm used by all models is the same. The time was significantly improved in the proposed mode. However, time difference for small datasets is not observed because of the network-communication delay. Table 1 shows the running time for all models for each dataset.

Table 1 Running time of all the models for each dataset (HH:MM: SS.ms)

Dataset	*Dataset size*	*Island Model*	*Neighborhood Model*	*Migration Model*	*GC PGA Model*
Abalone	*4177*	*01:08:33.60*	*01:46:23.20*	*01:36:19.28*	*00:24:12.32*
Breast C. W.	*569*	*00:04:35.52*	*00:05:56.85*	*00:05:03.30*	*00:04:19.75*
Cover type	*581012*	*512:15:45.3*	*216:45:55:10*	*211:12:50:38*	*10:25:45.20*
Ecoli	*336*	*00:00:24.40*	*00:00:16.30*	*00:00:14.95*	*00:00:12.95*
Haberman's	*306*	*00:01:03.55*	*00:00:56.74*	*00:00:51.39*	*00.00:45.53*
Image Segm.	*2310*	*00:29:28.28*	*00:38:59.28*	*00:32:19.18*	*00:04:00.15*
Ionosphere	*351*	*00:00:42.50*	*00:00:24.54*	*00:00:29.13*	*00:00:16.91*
Iris	*150*	*00:00:16.84*	*00:00:9.59*	*00:00:9.87*	*00:00:06.87*
ISOLET	*7797*	*09:01:12.36*	*07:15:36.15*	*06:56:12.15*	*01:32:15.92*
Car Eval	*1728*	*00:15:16.90*	*00:19:45.10*	*00:18:35.36*	*00:05:11.97*
Letter reco.	*20000*	*10:58:19.68*	*31:11:59.25*	*27:15:52.80*	*04:56:05.36*
Magic	*19020*	*08:19:28.82*	*25:49:22.50*	*21:12:22.56*	*03:46:28.25*
Pen digits	*7494*	*03:18:38.18*	*05:14:48.75*	*06:35:41.57*	*01:58:44.81*
SPECT -h	*187*	*00:00:06.95*	*00:00:04.51*	*00:00:04.46*	*00:04:23.25*
Dermatology	*366*	*00:01:29.75*	*00:01:34.46*	*00:01:04.41*	*00:23:37.68*
Lung cancer	*32*	*00:00:58.91*	*00:00:31.55*	*00:00:30.95*	*00:00:19.25*

Yeast	*1484*	*00:14:01.08*	*00:13:11.87*	*00:13:13.25*	*00:08:15.52*
M-graphy-m	*961*	*00:05:59.47*	*00:06:12.82*	*00:05:56.83*	*01:28:36.37*
Glass Id.	*214*	*00:00:53.69*	*00:00:45.69*	*00:00:46.35*	*00:00:41.24*
Musk	*476*	*00:02:01.20*	*00:01:56.26*	*00:01:51.60*	*00:12:56.39*
Page-blocks	*5473*	*02:31:36.27*	*03:24:31.56*	*03:04:31.21*	*00:39:01.68*
Pima-Id	*768*	*00:04:16.10*	*00:03:46.30*	*00:03:32.54*	*00:02:54.64*
Statlog-S	*4435*	*01:54:06.65*	*01:34:06.36*	*01:39:42.38*	*00:44:25.14*
Statlog-sh	*14500*	*16:51:29.35*	*26:09:33.12*	*23:56:12.60*	*03:11:56.57*
Zoo	*101*	*00:00:02.81*	*00:00:02.32*	*00:00:02.34*	*00:00:02.16*
Total		*567:40:47.89*	*320:46:50.17*	*304:54:30.84*	*30:15:48.76*
Average		*22:42:25.92*	*12:49:52.41*	*12:11:46.83*	*01:12:37.95*

Figure 1 illustrates the significant improvement in time of Global Control PGA over the other models for three large datasets. This choice was chosen because these large datasets show real differences over small ones that require long communication time. The figure also shows an improvement of the modified Island model over the original one. Notice that the time obviously decreased with the use of different models. It is clear that the best reduction in time has been achieved when using the proposed model.

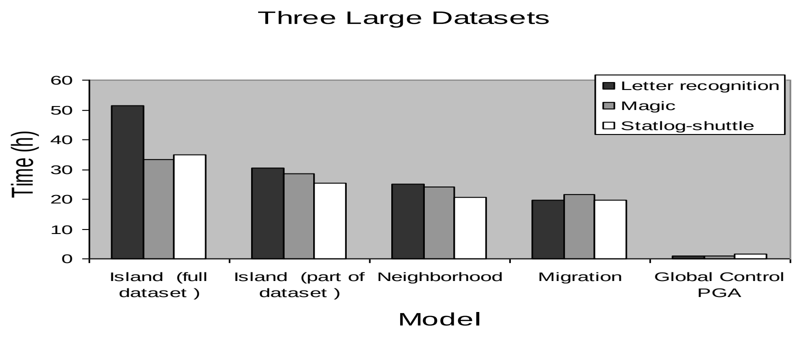

Figure 6 Running time for three large datasets using different models

Table 2 shows the ratio in dataset-size reduction. The results reveal an acceptable improvement in size reduction using the Global Control PGA model over all other models. The reason behind this is that the instance-reduction algorithm used in all models is the same algorithm. In the other hand, on average over all datasets the proposed model shows a superior performance over the other models in size reduction.

Table 2 Percentage of Size Reduction

Dataset name	*Island Model%*	*Neighborhood Model %*	*Migration Model %*	*Global Control PGA %*
Abalone	90.8	90.2	90.6	91.9
Breast C.W.	91.5	92.1	91.3	93.2
Cover type	78.84	85.5	88.0	87.6
Ecoli	82.9	83.3	82.7	85.2
Haberman's	91.9	92.9	91.2	94.8
Image Segm.	97.9	98.1	97.1	97.6
Ionosphere	85.8	88.2	86.0	87.8
Iris	94.8	94.6	94.7	96.1
ISOLET	94.6	94.1	94.3	95.2

Car Evaluation	85.8	83.9	84.1	88.3
Letter recog.	84.1	89.5	90.1	92.1
Magic	85.6	88.7	86.6	86.8
Pen digits	86.7	86.1	87.0	88.9
SPECT heart	87.7	86.7	88.1	89.1
Dermatology	90.1	91.6	92.6	94.6
Lung cancer	82.5	82.6	83.5	86.5
Yeast	85.8	86.6	85.8	87.8
mgraphy-masses	86.7	86.8	88.1	89.1
Glass Id	83.9	84.1	83.9	87.9
Musk	85.0	87.6	85.3	86.3
Page-blocks	92.4	93.6	92.9	97.9
Pima-Indians-d	85.6	87.1	86.7	87.7

Table 3 shows a summary of the average results for accuracy, size-reduction, and time. The accuracy of the resulting training set for all models are very close to each other. The Global Control PGA shows superior performance in both size-reduction and time over all models.

Table 3 Average Accuracy, Size Reduction, and Time for all models over all datasets

Model	*Accuracy %*	*Size reduction %*	*Time (H:M:S.ms)*
Modified Island Model	84.13	87.41	22:40:47.89
Neighborhood	84.44	87.65	12:49:52.41
Migration Model	84.86	87.77	12:11:46.83
Global Control PGA	84.46	90.22	01:12:37.95

Table 3 reveals that the accuracy for all the models are very close, this is because the same copy of the genetic algorithm was used for all the models. The table also shows a slight improvement of 3% in size reduction of Global control PGA over the other models. More importantly, the significant improvement was in time, about 95% reduction of time over Island model, 89% over Neighborhood model, and about 88% over Migration Model.

References

[1] Aha David W., (1992), “Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms,” International Journal of Man-Machine Studies, Vol. 36.

[2] Aha David W., Dennis K., Marc K., and Albert, 1991, "Instance-Based Learning Algorithms," Machine Learning, Vol. 6, pp. 37-66.

[3] Wilson D. Randall, and Tony R. Martinez, (1997) "Bias and the Probability of Generalization," In Proceedings of the 1997 International Conference on Intelligent Information Systems (IIS'97), pp. 108-114.

[4] Wilson D. Randall, and Tony R. Martinez,(2000), "Reduction Techniques for Exemplar-Based Learning Algorithms," Machine Learning vol. 38, no. 3, pp. 257-286.

[5] Wilson D. and Tony R. Martinez , (2000), "An Integrated Instance-Based Learning," Algorithm Computational Intelligence, vol. 16, no. 1, pp. 1-28.

[6] Enrique and José M., (1999), “An Analysis of Synchronous and Asynchronous Parallel Distributed Genetic Algorithms with Structured and Panmictic Islands," Parallel and Distributed Processing, J. Rolim et al. (eds.), Lecture Notes in Computer Science.

[7] Alba M. Troya, (1999), "A Survey of Parallel Distributed Genetic Algorithms," Complexity, vol. 4, no. 4, pp. 31-52.

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

[9] P.,(1968), "The condensed nearest neighbor rule," IEEE Transactions on Information Theory, vol. 14, No. 3, pp. 515–516.

[10] Ritter G., H., Wooddruff, S. Lowry, and T. Isenhour, (1975), "An algorithm for a selective nearest neighbor rule," IEEE Transactions on Information Theory.

[11] Muhlenbein H., Schomischm, and Born J., (1991), "The Parallel Genetic Algorithms Function Optimizer," Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann (San Mateo,CA).

[12] Hongmei H., S. Ondrej, A. Salagea, and E. Makinien, (2007), "Parallelization of genetic algorithms for the 2-page crossing number problem," Vol. 67, no. 2, February pp. 229-241, ISSN:0743-7315

[13] Cagatay, A. and Talay, (2005), "An Approach for Eye Detection Using Parallel Genetic Algorithm," Springer Berlin / Heidelberg Volume 3516/2005

[14] Sprin Y., Goldberg, D., Yassine A. and Chen, Y., (2003), "A genetic algorithm design inspired by organizational theory," Artificial Neural Networks in Engineering (ANNIE 2003), pp. 327–332. Germany, Berlin, pp. 126–137.

[15] Kotsiantis, S. B., 2007, "Supervised Machine Learning: A Review of Classification Techniques," Informatica, No. 31, pp. 249-268.

[16] Dirk S., and H. Muhlenbein, (1996), "Adaptation of population sizes by competing subpopulations," Proceeding of the IEEE international conference on evolutionary computation, pp. 330-335.

[17] UCI Repository of Machine Learning. Irvine, CA: University of California Irvine, Department of Information and Computer Science, (Date of last visit: February 20, 2011) from: http://archive.ics.uci.edu/ml/index.html.

[18] Hughes, E. J., (2005), "Evolutionary many-objective optimization: many once or one many?," Proc. of 2005 IEEE Congress on Evolutionary Computation (Edinburgh, UK, September 2-5, 2005), pp. 222-227.

Author

Dr Tahseen Al-Ramadin (tahsen@ahu.edu.jo) received his BSc in computer science in 1987 from Yarmouk University, Irbid-Jordan. He perused his MSc and PhD. in 2003 and 2009, respectively, both in Computer Information Systems from Arab Academy, Amman, Jordan. He is a Professor in the faculty of information technology of the Al-Hussein Bin Talal University, Ma'an, Jordan.