Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 46 (2015) 276 – 284

Procedia Computer Science

International Conference on Information and Communication Technologies (ICICT 2014)

Discovery of Classification Rules using Distributed Genetic Algorithm

Priyanka Sharma [a,*], Saroj[b]

[a]*Chaudhary Devi Lal University, Sirsa,125055,India*
[b]*Guru Jambheshwar University of Science & Technology, Hisar,125001,India*

Abstract

This paper presents a distributed genetic algorithm for the discovery of classification rules. Population is contained in the form of interconnected demes. The local selection and reproduction mechanism is used to evolve the species within demes, and diversity is enhanced by migrating rules among some of the selected demes. Subsumption operator has been finally applied to reduce the complexity of the rule set discovered. The effectiveness of the proposed distributed genetic algorithm for discovering classification rules is evaluated by comparing the results with traditional crowding GA on 10 datasets from the UCI and KEEL repository. The results confirm that the distributed GA discover classification rules with significantly higher predictive accuracy. The influence of migration operator is also analysed with respect to migration rate.

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)

*Keywords:*Distributed GA; Classification rules; Parallel GA.

1. Introduction

The phenomenal growth of data in the recent decades has necessitated data mining algorithms to extract useful and actionable knowledge from large datasets. Classification[13] is one of the most widely used data mining techniques to build a model that describes and distinguishes data classes in a manner to be used to predict the class of unseen instances i.e. instances whose class labels are unknown. Genetic algorithms[19] are robust and adaptive search algorithms that perform global search in the solution space. As the possible candidate rule space is large even for the

* Priyanka Sharma. Tel . +91 9466695201
E-mail Id.: pinki.sharma2912@gmail.com

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)
doi:10.1016/j.procs.2015.02.021

moderate sized datasets, GAs have been extensively used for discovering classification rule in the form of *If-Then* symbolic rules. Although GAs have been able to successfully discover accurate, interesting and comprehensible rules[8,10,11,20], achieving a right balance of exploration and exploitation is a precondition for any GA to avoid premature convergence to a sub-optimal solution. One way to adjust exploration-exploitation balance is to optimize GA parameters. Researchers have also suggested several selection, scaling and diversity mechanisms to achieve a good balance of exploration and exploitation[18]. However, these solutions could not completely eliminate the problem of premature convergence. Therefore, Parallel or Distributed Genetic Algorithms[2,3] (DGAs) followed as the most important method to address the problem of simple GA converging to local optimal solutions at times. A DGA is also known as Island model or a Coarse grained GA. The DGAs divide the population into subpopulations and preserves the diversity due to semi-isolation of theses sub-populations during evolution. These sub-population evolves independently and simultaneously. Periodically, a migration operator exchanges some better performing individuals among the sub-populations to build on the promising search directions. We have not come across many contribution where DGAs have been employed for discovery of classification rules. Therefore, we propose a distributed genetic algorithm in which a crowding GA is applied within demes, a migration operator in between demes after a specified interval and finally a subsumption operator is applied to subsume the rules of lesser coverage. DGAs[9] are capable of producing solutions with higher efficiency (in terms of time) and efficacy (in terms of better quality solutions). A time advantage is achieved when subpopulations contained in DGA are evolved on multiple processors in parallel. In this work we are using DGA for classification rule discovery to get accurate and comprehensible rules and the proposed approach is simulated on a single processor. Effectiveness of the proposed algorithm is evaluated by comparing the results with traditional crowding GA on 10 datasets from the UCI machine learning repository[5] and KEEL[4] repository. The results confirm that distributed GA achieves a higher predictive accuracy across all the data sets. The performance of DGA is also analysed with respect to migration rate, frequency of migration and type of rules that migrate.

2. Literature Review

Many data mining algorithms have to cope with scalability problem due to large datasets and curse of dimensionality problem. The search space for candidate rules for discovery of classification rules grows exponential with the number of attributes. Therefore, several researchers have applied genetic algorithms to address the above challenges in the domain of data mining[6,17]. Owing to the problems of premature convergence and long running times of simple GAs, the parallel versions of genetic algorithms came into existence. Though PGAs are specifically suitable to address data mining problems, we have come across only a limited number of research papers in this domain.

Fitness function is usually computationally very expensive for rule mining problems and therefore a Master slave architecture[17] has been implemented to distribute the fitness computations among several processors. The effective performance of this architecture is then validated using standard test bed functions and a set of classification data mining problems. Parallel GAs[2] not only increase efficiency in terms of time but also lead to superior optimization performance. A Parallel GA, even when implemented on a single processor, is capable to give better optimized solutions. This has been proved by applying several kinds of Distributed Genetic Algorithms- A class of PGAs- in the domain of function optimization[1,14,16]. A Self-Adaptive Migration Model GA[22] (SAMGA) has been proposed for discovering classification rules. In this paper parameters like population size, the number of points of crossover and mutation rate are adaptively fixed for every successive generation. Variable sized demes, and the replacement of demes by migrants every generation is used in the Island model. Further, the migration of individuals between populations is decided dynamically. DGAs have also been implemented for discovering rules with exceptions and hierarchies[6,7,21].

3. The Proposed Approach

DGAs are not just parallel versions of sequential genetic algorithms. In fact, they actually reach the ideal goal of having a parallel algorithm whose behavior is better than the sum of the separate behaviors of its component sub-algorithms. The Proposed DGA approach makes local search intra-demes/ intra-islands and global search in inter-

demes/ inter-islands. Isolated subpopulations in form of islands evolve locally for few specified number of generations, and then migration of rules takes place in-between subpopulations. Frequency of migration and migration rate are important factors that influence the performance of any DGA.

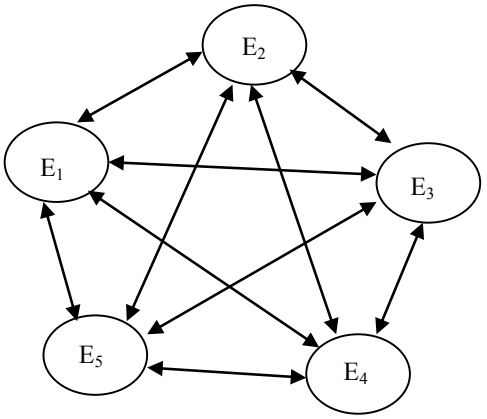

Fig. 1. The Proposed DGA.

Our DGA contains five demes/ islands E_1, E_2, E_3, E_4 and E_5 as shown in Fig. 1. The demes are selected in pairs and then rules are exchanged between them subject to the improvement in the average accuracy of the rule sets contained in the participating demes.

3.1. Population Initialization

Population Initialization: The subpopulations are initialized by generating random rules. Fixed length integer encoding has been used for genotype representation of individuals. Phenotype is represented in the form of '*If* P *Then* D' rules; where P is premise/antecedent and is a conjunction of attribute value pairs. Decision/consequent D is a single term that contains the value for the goal attribute. We have taken Iris dataset from UCI repository to explain initialization, the operators and fitness measure employed in our work. Iris data set contains continuous data. We have discretized the data using Weka. All the attributes contain three values after discretization. A table has been shown below representing all the attributes of the Iris dataset for the convenience of a reader to understand it better.

Table 1.Description of Iris Dataset.

Locus	Attributes	Value	Alleles
0	Petal length	High, Medium, Small	'1', '2', '3'
1	Petal width	Large, Medium, Small	'1', '2', '3'
2	Sepal length	High, Medium, Small	'1', '2', '3'
3	Sepal width	Large, Medium, Small	'1', '2', '3'
4	Iris	Setosa,Versicolor,Virginica	'1', '2', '3'

Given three class values 'setosa', 'versicolor' and 'virginica' encoded as '1', '2' and '3', a sample rule set is shown in Fig. 2. It shows encoded rule set (genotype), and its respective decoded rule set (phenotype).

After population initialization, sub-populations evolve locally for a specified interval and then migration among subpopulations takes place. The process continues till stopping criteria is not reached. Finally, the subsumption operator is applied to reduce the complexity of the rule set collected from all demes.

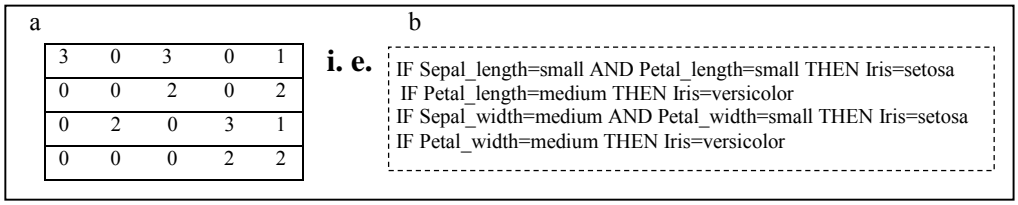

Fig. 2. Representation: (a) Genotype; (b) Phenotype.

3.2. Intra-island/ local evolution

Crowding genetic algorithm has been applied to evolve the subpopulations E_1, E_2, E_3, E_4 and E_5 locally. All the subpopulations are evolved in isolation simulating parallel processing. The fitness measure used, selection and recombination operators are explained below.

3.2.1 Fitness function

The fitness function evaluates the quality of each rule (chromosome) in the population. Before we can define the fitness function, it is necessary to recall a few basic concepts on classification accuracy measures of a rule. When using a rule for classifying an example, depending on the class predicted by that rule and on the true class of the example, four different types of results are observed for the prediction, as follows:

- True Positive (TP) – positive class correctly predicted by the rule i.e. P ^ D;
- False Positive (FP) – negative class incorrectly predicted by the rule i.e. P ^ ~D;
- True Negative (TN) - negative class correctly predicted by the rule i.e. ~P ^ ~D;
- False Negative (FN) - negative class incorrectly predicted by the rule i.e. ~P ^ D;

Here, P is the premise and D is the decision part of a rule.

Our fitness function combines two measures, sensitivity and specificity, defined as follows:

$$Sensitivity = \frac{|TP|}{|TP + FN|}$$

$$Specificity = \frac{|TN|}{|TN + FP|}$$

Finally, the fitness function used by our classification is defined as the product of these two measures and named as *Rrule_score*, i.e.:

$$Rule\ Score = Sensitivity \times Specificity$$

Therefore, the goal of our evolutionary mechanism is to maximize both the *Sensitivity* and the *Specificity* at the same time. However, if a rule happens to contain don'care states only, it has been assigned a fitness equal to zero.

3.2.2. Genetic operators

- *Selection operator*
  Roulette wheel selection has been used for selecting individuals to take part in the further evolutionary process by generating new and better offspring.
- *Crossover operator*
  Traditional one-point crossover operator has been used to recombine the individuals generating new genetic material. A random site is selected, and then corresponding values are exchanged generating new individuals. When two individuals belonging to different classes recombine, they most often produce worst offspring because these belong to different species in the population. Therefore, a crossover is limited to the individuals belonging to the same class.

- *Mutation operator*
  Mutation operator is used to maintain diversity in the population. An adaptive mutation has been applied to make the evolution better so that we don't lose better individuals towards the final runs. Initially, mutation probability is set to 0.1, probability gets updated depending upon the accuracy of the demes with respect to previous generation accuracy.

$$P_{mut} = \begin{cases} 0.1; & if\ Accuracy < \Pr ev\_gen\_accuracy \\ \dfrac{(Accuracy - \Pr ev\_gen\_accuracy)}{Accuracy} & otherwise \end{cases}$$

Mutation operator applied may specialize (a don't care state i.e. 0 mutated to any value) or generalize (attribute value mutated to a don't care state) a candidate rule by inserting or removing conditional clauses in the antecedent part of the rule.

3.2.3 Crowding

A Crowding technique has been employed to maintain diversity in the subpopulation and to handle problem of convergence of the whole population to the single best rule. Convergence to a single best rule makes the GA inefficient, as the GA need to run multiple times for getting multiple rules. In crowding technique, we maintain an overlapping population instead of generating a whole new population in every generation and the offspring produced through GA operators replace the worst but similar individuals to maintain the diverse species (rules in our case) in the GA population . We have used the following mechanism to measure the similarity between the individuals of the GA population and offspring:

$$Similarity = \frac{Int}{Uni}$$
$$Int = |N_1 \cap N_2|$$
$$Uni = |N_1 \cup N_2|$$

Where, N_1 is the number of instances covered by an offspring rule and N_2 is the number of the instances covered by a worst rule. *'Int'* represents common instances covered by the offspring and the worst rules and '*Uni*' represents total instances covered by the offspring and the worst rules collectively. The similarity will always take values between 0 and 1. Higher the value is, more similar the two rules are. The offspring rule replaces the worst rule with maximum similarity out of a number of the randomly selected worst rules from the GA population. The reader may refer to the reference[12] for the details of the crowding technique.

3.3. Inter-island/ global evolution

Inter-island or inter-deme evolution of genetic materials is done among subpopulations to achieve a better balance between exploration and exploitation. Accuracy of the subpopulation is evaluated as a whole to know its quality, and then exchange and migration of rules take place among the demes.

3.3.1 Accuracy

We need to compute accuracy of a sub-population (rule set contained in a deme) for inter-deme evolution. This choice of fitness function is of great importance as it leads the search towards globally optimal rule set. The fitness function for inter-deme evolution is given below.

$$Accuracy = \frac{|TP + TN|}{N}$$

where, N is the total number of instances in the Dataset.

3.3.2 Migration operator

Multiple pairs of demes are selected randomly and their fitness is calculated. Pairs of demes selected are either equal to the number of islands or less than the number, depending on the pair selection, as if same deme is

selected twice in a pair then it will be discarded. As shown in Fig.1, multiple pairs are selected, it should be noted that not all the combinations are necessarily selected but only a few are randomly chosen. The process of exchanging individuals between the selected demes has been shown with the help of example in Fig. 3.
After exchanging the N best individuals between the selected pairs, the average fitness of new demes are recomputed and compared to the average fitness's of original selected demes. If the average of fitness's of parent subpopulations is less than the offspring's average fitness, then the parent subpopulations in the population are replaced by offspring subpopulations otherwise there is no change in the population. In the example, Rule1 of subpop1 has been exchanged with Rule1 of subpop2. Sum of accuracy's of parent subpopulations computes to 0.72 using inter-deme fitness evaluation function which is less than sum of accuracy's of offspring organizations which evaluates to 0.865. Therefore, exchange of rules takes place among subpopulations.

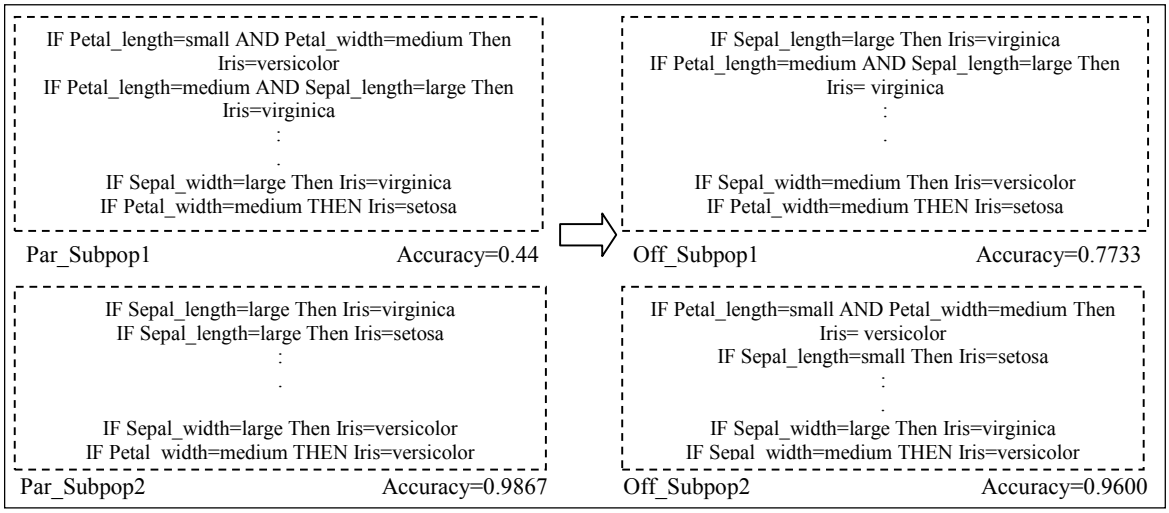

Fig. 3. Migration operator.

3.4. Subsumption operator

The subsumption operator is used as a post-processing operator after the end of the evolution process. The rules are collected from all the demes into a single pop and then subsumption operator is used to remove those rules from the subpopulation which cover the subset of data instances covered by any other rule in the population. This operator reduces complexity by removing duplicacy and redundancy from the overall rule set discovered. The subsumption operator is depicted in the Fig. 4.

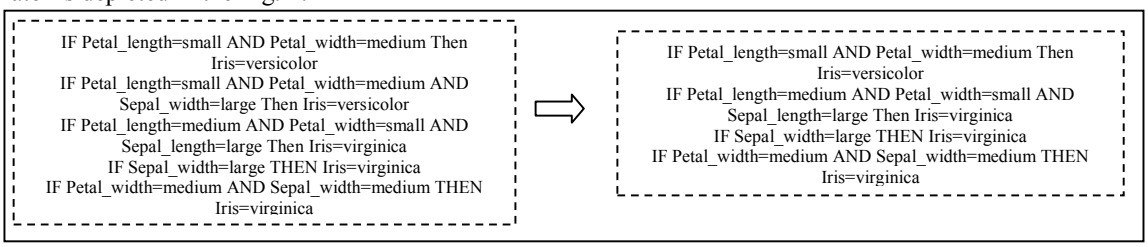

Fig 4. Subsumption operator.

Here, data instances covered by the rule '*IF Petal_length=small AND Petal_width=medium AND Sepal_width=large Then Iris=versicolor*' is subset of the data instances covered by the rule 'IF Sepal_width=large Then Iris=versicolor'. Thus, the rule '*IF Petal_length=small AND Petal_width=medium AND Sepal_width=large Then Iris=versicolor*' is removed from the population because it gets subsumed by the general rule 'IF Sepal_width=large Then Iris=versicolor'.

4. Experimental Design Results

We have validated the proposed distributed GA approach on ten datasets taken from the KEEL (a software tool to assess evolutionary algorithms in data mining problems) and UCI machine learning repository. All the datasets are summarized in Table 2. We have used twofold cross validation for the datasets that contains less than 700 hundred instances and a tenfold cross validation for the datasets that contain more than 700 hundred instances. The parameter setting is an important task in GA based approaches. After doing some experimentation for tuning the parameters, we arrived on the parameter values as given in Table 3.

Table 2. Datasets used in Experiments.

Datasets	#Instances	#Attributes	#Classes	Test Method
Mushroom	5644	23	2	10-CV
Car	1728	7	4	10-CV
Vote	435	17	2	2-CV
Poker Hand	25010	11	10	10-CV
Iris	150	5	3	2-CV
Zoo	101	17	7	2-CV
Breast Cancer Wisconsin (BCW)	683	10	2	2-CV
Breast Cancer	277	10	2	2-CV
Nursery	12960	9	5	10-CV
Tic-Tac-Toe	958	10	2	10-CV

Table 3. Optimized Parameters of DGA.

Parameter	value
Population size for Crowding GA	400
Number of islands	5
Deme Size	80
Selection algorithm	Roulette wheel
Maximum number of generation	150
Mutation probability	Adaptive with initial as (0.1)
Crossover probability	0.6
Crowding Factor	4
Frequency of migration	5
Number of migrations	8

Table 4. Comparision of DGA with Crowding GA.

Datasets	Crowding GA	DGA
Mushroom	0.7592 ±0.0870	0.9816 ±0.0126
Car	0.5735 ±0.0367	0.7069 ±0.0576
Vote	0.9508 ±0.0236	0.9863 ±0.0064
Poker Hand	0.4604 ±0.1221	0.6636 ±0.0707
Iris	0.9453 ±0.0183	0.9867 ±0.0
Zoo	0.8453 ±0.0355	0.9503 ±0.0130
Breast Cancer Wisconsin	0.7147 ±0.0404	0.8902 ±0.0230
Breast Cancer	0.7797 ±0.0727	0.9168 ±0.0226
Nursery	0.6762 ±0.0698	0.7682 ±0.0513
Tic-Tac-Toe	0.7059 ±0.0367	0.9301 ±0.0273

A Comparison of predictive accuracies obtained on test sets using a simple crowding GA and the proposed deme GA is presented in Table 4. It is clear from this table that the deme GA performs better as compared to the simple crowding GA across all the ten data sets. We have also plotted the graphs for the four datasets (Mushroom, Car, Tic-tec_toe, Nursery) for accuracy over ten runs of the simple crowding GA and the proposed deme GA in Fig. 5. This shows that DGA performs consistently better. The better performance can be attributed to the fact that the deme GA first optimizes the list of best rules and then inter-demes migrations help to optimize to the best set of rules. The

simple GA takes care of attribute interaction only i.e how two or more attributes together can influence the class value. The DGA suffices for attribute as well as rule interactions i.e. how two or more number of rules work better or worse in combinations. Thus, DGA generates the better rule list.

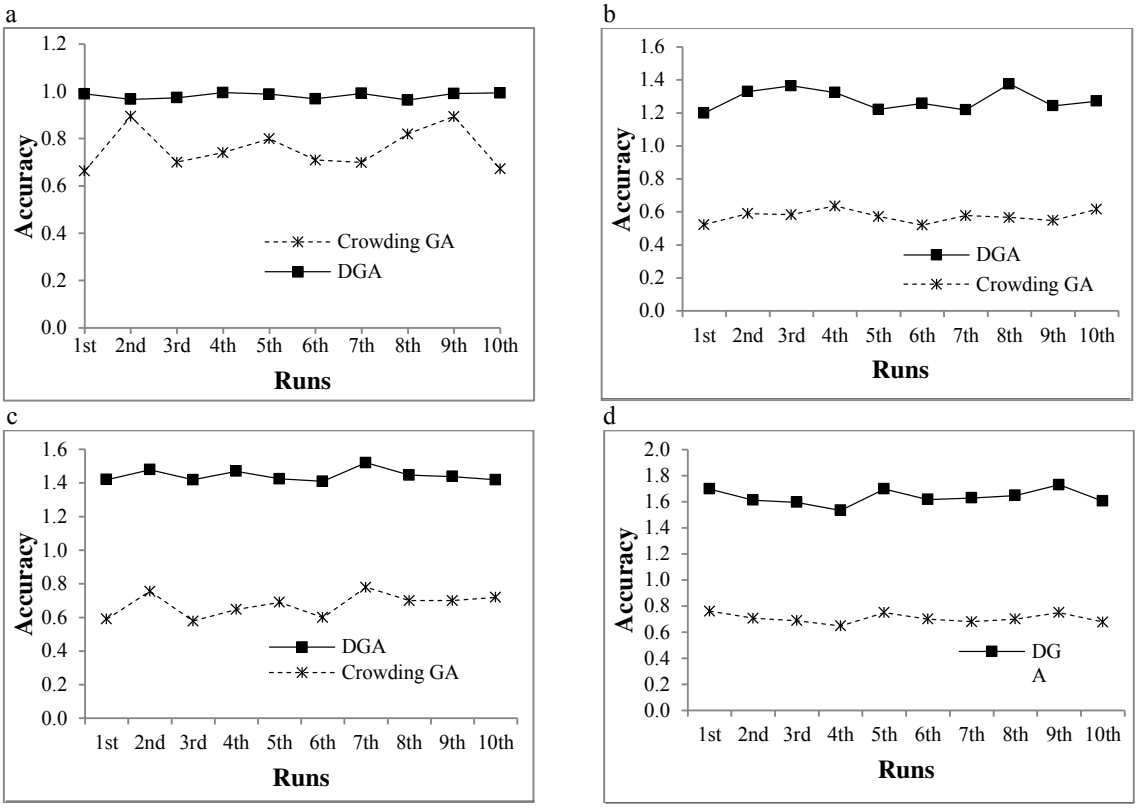

Fig. 5. Accuracy in 10 runs on datasets: (a) Mushroom; (b) Car; (c) Nursery; (d) Tic-Tac-Toe.

Table 5. DGA Accuracy with different number of Migrations and Exchanges on various Datasets.

Datasets	Exchg=2	Exchg=4	Exchg=8	Exchg=12
Mushroom	0.7810 ±0.0170	0.8620 ±0.0130	0.9816 ±0.0126	0.8012 ±0.0154
Car	0.5815 ±0.0125	0.6823 ±0.0457	0.7069 ±0.0576	0.6082 ±0.0743
Vote	0.9320 ±0.0315	0.9820 ±01430	0.9863 ±0.0064	0.9210 ±0.0435
Poker Hand	0.5212 ±0.3210	0.5345 ±0.0231	0.6636 ±0.0707	0.5312 ±0.2510
Iris	0.9613 ±0.0175	0.9820 ±0.1100	0.9867 ±0.0	0.9667 ±0.0471
Zoo	0.8231 ±0.0430	0.8716 ±0.2760	0.9503 ±0.0130	0.8418 ±0.0824
BCW	0.7617 ±0.0124	0.8231 ±0.0143	0.8902 ±0.0230	0.8440 ±0.6210
Breast Cancer	0.8112 ±0.0136	0.8534 ±0.0341	0.9168 ±0.0226	0.8445 ±0.0978
Nursery	0.7012 ±0.0281	0.7680 ±0.0371	0.7682 ±0.0513	0.6971 ±0.0382
Tic-Tac-Toe	0.7212 ±0.0423	0.8214 ±0.628	0.9301 ±0.0273	0.7744 ±0.1331

Further, we have analysed the impact of migration rate on the performance of DGA and the results at different migration rates are shown in the Table 5. It can be concluded from the experiments that if migration rate is very low,

then DGA works like crowding GA. The performance of DGA also decreases if migration rate is increased beyond a limit. This is due to the fact diversity decreases at higher migration rates. An optimal migration rate has to be found for the optimal performance of a DGA. The optimal migration rate for the proposed DGA is 8 for all the data sets.

5. Conclusions

We have proposed a DGAs for discovery of accurate classification rules. In the proposed DGA approach the population is divided into sub-populations and evolved simultaneously in separation for localized competition. Migration of individuals takes place by inter-deme exchange of individuals. The proposed approach has been validated on 10 data sets. The DGA approach has been able to discover rule sets with significantly high accuracy as compared to a simple crowding GA. The approach optimizes the list of best rule within demes and the inter-deme migration brings about the best rule list. However, the performance of DGA depends on migration rate. There is an optimal migration rate where the DGA gives the best results. We have used homogeneous DGAs i.e. a single genetic algorithm with same parameter settings is applied for intra-deme evolution. Heterogeneous and Hierarchical DGAs have shown promising results for optimizing numeric functions[15,16] and these can be tried in the domain of rule mining.

References

1. Alba, E., Nebro, A.J., Troya, J.M. Heterogeneous computing and parallel genetic algorithms. Journal of Parallel and Distributed Computing 2002; 62: 1362–1385.
2. Alba, E., Tomassini, M. Parallelism and evolutionary algorithms. IEEE Transactions on Evolutionary Computation, 2002; 6: 443–462.
3. Alba, E., Troya, J.M. A survey of parallel distributed genetic algorithms. Complexity 1999, 4: 31–52.
4. Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. J. Mult.-Valued Log. Soft Computing 17, 2011.
5. Asuncion, A., Newman, D. UCI machine learning repository, 2007.
6. Bharadwaj, K.K., Saroj. Parallel Genetic algorithm approach to automated discovery of hierarchical production rules, In: Applications of Soft Computing. Springer, 2009; 58: 327–336.
7. Bharadwaj, K.K., Saroj. A parallel genetic programming based intelligent miner for discovery of censored production rules with fuzzy hierarchy. Expert Systems with Applications 2010; 37: 4601–4610.
8. Fidelis, M.V., Lopes, H.S., Freitas, A.A. Discovering comprehensible classification rules with a genetic algorithm, In: Evolutionary Computation, IEEE, Proceedings of the Congress, 2000; 1: 805–810.
9. Freitas, A.A. A survey of parallel data mining, In: Proceedings of 2nd International Conference on the Practical Applications of Knowledge Discovery and Data Mining. The Practical Application Company, 1998; 287–300.
10. Freitas, A.A. A Review of evolutionary Algorithms for Data Mining, In: Maimon, O., Rokach, L. (Editors.), Soft Computing for Knowledge Discovery and Data Mining. Springer US, 2008; 79–111.
11. Gopalan, J., Alhajj, R., Barker, K. Discovering Accurate and Interesting Classification Rules Using Genetic Algorithm., In: Proceedings of the International Conference on Data Mining, DMIN 2006; 389–395.
12. Golberg, D. E. *Genetic Algorithms*, Pearson Education India, 2006.
13. Han, J., Kamber, M., Pei, J. *Data mining: concepts and techniques*.2nd ed. Canada: Morgan kaufmann, 2006.
14. Herrera, F., Lozano, M. Gradual distributed real-coded genetic algorithms. IEEE Transactions on Evolutionary Computation, 2000; 4: 43–63.
15. Herrera, F., Lozano, M., Moraga, C. Hybrid distributed real-coded genetic algorithms, In: Eiben, A., Bäck, T., Schoenauer, M., Schwefel, H.-P. (Editors), Parallel Problem Solving from Nature — PPSN V, Lecture Notes in Computer Science. Springer Berlin Heidelberg,1998; 603–612..
16. Herrera, F., Lozano, M., Moraga, C. Hierarchical distributed genetic algorithms. International Journal of Intelligent System 1999; 14: 1099–1121.
17. Hong, T.-P., Lee, Y.-C., Wu, M.-T. Using the master-slave parallel architecture for genetic-fuzzy data mining, In: IEEE International Conference on Systems, Man and Cybernetics 2005; 4: 3232–3237.
18. Kapila, Saroj, Kumar, D., Kanika. A genetic algorithm with entropy based initial bias for automated rule mining, In: IEEE International Conference on Computer and Communication Technology (ICCCT), 2010; 491–495.
19. Michalewicz, Z. *Genetic algorithms+ data structures= evolution programs*. springer, 1996.
20. Noda, E., Freitas, A.A., Lopes, H.S. Discovering interesting prediction rules with a genetic algorithm, In: IEEE Proceedings of the Congress on Evolutionary Computation, CEC 99, 1999; 2: 1322–1329.
21. Saroj, S., Bharadwaj, K.K. A parallel genetic algorithm approach for automated discovery of censored production rules, In: Proceedings of the 25th Conference on Artificial Intelligence and Applications. ACTA Press, 2007; 435–441.
22. Srinivasa, K.G., Venugopal, K.R., Patnaik, L.M. A self-adaptive migration model genetic algorithm for data mining applications. Information Science 2007; 177: 4295–4313.