Getting the Best Rules using Genetic Algorithm Variants

L. M. R. J Lobo[1] R. S. Bichkar[2]
[1]Research Scholar & Associate Professor [2]Professor
[1]Department of Computer Science & Engineering [2]Department of Electronics & Telecommunication Engineering
[1]SGGS Institute of Engineering & Technology, Nanded India & Walchand Institute of Technology, Solapur India [2]G. H. Raisoni College of Engineering and Management, Pune India

Abstract— A large amount of transaction data gets stored on storage devices every day. To convert this huge amount of data into useful information we have proposed some data mining techniques which represent datasets (with the help of rules) to a maximum extent. A popular method to generate such rules is Association Rule Mining. The popular rule mining algorithm named Apriori algorithm can be used to generate these rules. However the efficiency and optimized set of rules can be achieved using Genetic algorithms and its variants. The present paper elaborates on a system developed to find all the possible optimized rules from the data set generated by Apriori algorithm using Genetic Algorithm and using its variants to improve the performance of the system by speeding up the time required for getting the results after scanning huge datasets. Experimentally a speedup of 1.67 using Parallel Genetic Algorithms and a speedup of 1.76 using Hierarchical Genetic Algorithms respectively as compared to using Simple Genetic Algorithms is generated by the system proposed.

Key words: Association Rule Mining, Genetic Algorithm

I. INTRODUCTION

In developing of an application for an industrial problem, researchers try to identify the most important objects and factors that will affect the output generated by them. This is dependent on the parameters which are applied to functions developed as main internal parameters and external interactions. The goal is however to find the 'best' [1] choice.

Association rule mining is a method, used to detect all subset of items which occur frequently and the relationship between them. Association rules are in the form of if-then statements that help uncover relationships between seemingly independent data in a relational database or other information systems [2]. Genetic Algorithm (GA) is an evolution based search technique. It involves processes of natural evolution given by Darwin's Theory. They use principle techniques of inheritance such as mutation, selection and crossover [3].

The Parallel Genetic Algorithm has proved to be computationally more efficient. The modifications with respect to Genetic algorithms is creating individual chromosomes to be 2-Dimensional, permitting mating of individuals to happen independently in its neighbourhood and improving of fitness during its lifetime. Hierarchical Genetic Algorithm is a kind of Genetic Algorithm to solve problems that have hierarchical structures. The goal of the Hierarchical Genetic Algorithms is governed by nodes which are configured hierarchically abiding a known order of placement for computations to be carried out.

II. RELATED WORK

Genetic Algorithm is discussed in [4]. In a genetic algorithm, an initial population which consists of chromosomes or possible candidate solutions are converted to generate an optimization better solution. Each chromosome (candidate solution) has a set of properties. The solutions are represented in a structure of strings of binary alphabet 0s and 1s, other encodings are also possible. The procedure of evolution usually starts with an initial population selected randomly [5]. In each generation that is a sub-lifetime or intermittent time, the fitness of every individual in the population is evaluated.

Parallel Genetic algorithm is discussed in [6], [7]. Genetic algorithms are well-suited for parallelization [8]. Genetic Algorithms are used to solve difficult problems in many disciplines. Unfortunately, they consume a lot of computation load and memory [9]. The basic idea behind most parallel programs is to divide a large problem into smaller tasks. These tasks are solved simultaneously on multiple processors. The Parallel Genetic Algorithms can be further classified [10]. A parallel approach (Grid) is discussed in [11]. In which each individual process performs its own selection. An efficient method based on Genetic Algorithms is developed in [12] to solve the multiprocessor Scheduling problem. Lisboa [13] introduced a parallel approach as a variant of the Genetic Algorithm, called Parallel Genetic Algorithm with Social Interaction (PSIGA).

Hierarchical Genetic Algorithm is discussed in [14]. The Hierarchical Genetic Algorithm is a kind of Genetic Algorithms applicable to solve problems that have a hierarchical structure. Anna and Maciej [15] used the hierarchical chromosome based Genetic Algorithm for optimal selection of the initial mesh. Gulsen and Smith [16] described a Hierarchical Genetic Algorithm framework for identifying closed form functions for multi-variable data sets. The hierarchy was assisted with an upper Genetic Algorithms that searched for appropriate functional forms. Yu et al. [17] proposed a genetic algorithm aided optimization scheme for designing the organization of hierarchical multi-agent systems. They introduced the Hierarchical Genetic Algorithm, in which hierarchical crossover with a repair strategy and mutation of small perturbation was used. In a Classification Rule Mining model with Genetic Algorithms the basic task [18] to be performed for using a parallel algorithm approach was illustrated.

III. METHODOLOGY

Apriori Algorithm is first implemented to generate association rules. Here we acquire all the possible rules. Genetic Algorithms is then applied on these rules to get the most optimised ones. We repeat the same for a parallel

configuration approach of Genetic algorithm explained in the subsections below and the hierarchical configuration approach of Genetic Algorithms.

A. Getting optimistic rules combining Apriori Algorithm and Genetic Algorithm

Optimistic Rules generation [19] using Genetic Algorithm is shown in Fig. 1. Finding interesting rules that are optimized are our goal. These rules will represent a data set to its optimal extent and is a necessity in most market basket analysis systems. The large datasets and the associated rules make the problem NP hard. We choose genetic Algorithms to find a solution since it works in generations and caters to reducibility of the problem.

1) Rule generation using Apriori Algorithm

The Input dataset is analysed and saved into a proprietary format. Then the transaction database from the input dataset is created. The user is then asked to input the values of threshold support and confidence. This is followed by Construction of the items database. Apriori Algorithm is then applied for generating rules.

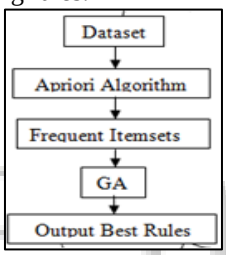

Fig. 1: Optimistic Rule generation using Genetic algorithm

The method followed begins by generating frequent item sets with just one item and to recursively generate frequent item sets with two items, then 3 item sets and so on, until we have generated frequent item sets of all sizes. The support and confidence for each set is then calculated. Now depending on the declared threshold support by the user we select only those rules that can be built supporting this support value then we refer to threshold confidence value given by the user we select only those rules that fit into this confidence condition. These rules are then saved in the rule database.

2) Application of Genetic Algorithm

After rule acquisition Genetic algorithm can be applied. The method used differs from a standard Genetic Algorithm in several crucial respects, including handling variable-length chromosomes that is association rules, which contain n-place predicates. Genetic Algorithms parameters are taken from the user. In order to efficiently explore the space of association rules it uses macro mutations as a generalization and specialization operators. Every individual (perspective solution) used in the Genetic algorithm is a chromosome. Association rules generated by Apriori algorithm are the chromosomes, whereas the antecedent and consequent in the rule form the genes in the chromosomes. Attributes of the dataset in a rule represent genes.

In the rule population, chromosomes are encoded to represent a single rule that is variable in length having symbolic structures that is association rules that may contain n-place predicates (n > 0). Binary encoding using Michigan approach, where each rule is encoded as a single chromosome is used. Two bits are used to encode the attribute. The antecedent part is denoted by 00, consequent part by 11 whereas 01 or 10 represent absence of any attributes in the rule.

For example consider a dataset having attributes {bread, butter, jam, milk, beer} and the rule

R1: if bread Λ jam → milk

R1 is encoded as 00 01 00 11 01.

The genetic operators, such as crossover and mutation, are then applied to reproduce offspring for the new main population. The selected parents can be crossed over with user specified crossover rate. Single point crossover is used and the crossover point is generated randomly. Single bit flip mutation with user specified mutation rate is used.

Rule fitness depends primarily on Confidence and support and optionally on parsimony reward based on data coverage. For each proposed rule, Confidence and Support are calculated based on the available Confidence = 0 or if it covers new data it gets Fitness = 0, otherwise the distance of its Confidence and Support from the target values is determined, using a Quasi Euclidean distance.

$$\text{Fitness} = (C - Tc)^2 + (S - Ts)^2 \qquad (1)$$

Where, C is confidence calculated by the system, Tc is threshold confidence given by the user, S is support calculated by the system and Ts is threshold support given by the user.

The resulting value is normalized as a percentage, it is adjusted by rewarding by a small constant for each conjunct by which the maximum antecedent size exceeds the rules antecedent size so that

$$\text{Fitness} = \text{Fitness} - (\max As - As) * R \qquad (2)$$

Where, as is antecedent size and R is award attached to good computations.

In classification tasks, fitness value is further adjusted by taking the data coverage from the rule into account. The rules are maintained in descending order of fitness. For rules covering same data, only the rule with the highest Confidence and Support get credited for these data. This encourages an extensive exploration of the data set. Data coverage refers to perfect and estimated coverage. These measures promote default hierarchy.

Initial population is selected for Genetic Algorithm. Fitness function is based on confidence and support values of association rules. If a rules confidence = 0 or if it covers no new data, it gets 0 fitness. Thus fitness depends primarily on confidence and support values. After evaluating fitness function Genetic Algorithms operators (crossover and mutation) are applied. If the new individuals are better than their parents then they form the new population and take part in reproduction. Again this entire procedure is repeated until the best rules are obtained. This iterative process ends with set of best rules.

B. Optimistic rule generation using Parallel Genetic Algorithm

This is based on master-slave Parallel Genetic Algorithm [20], [21]. Here master generates the initial population, distributes population to number of connected Slaves and collects the best rules from different servers (slaves) and finally gives the best rules to the user. Thus the co-evolution can be achieved. This is shown in Fig. 2.

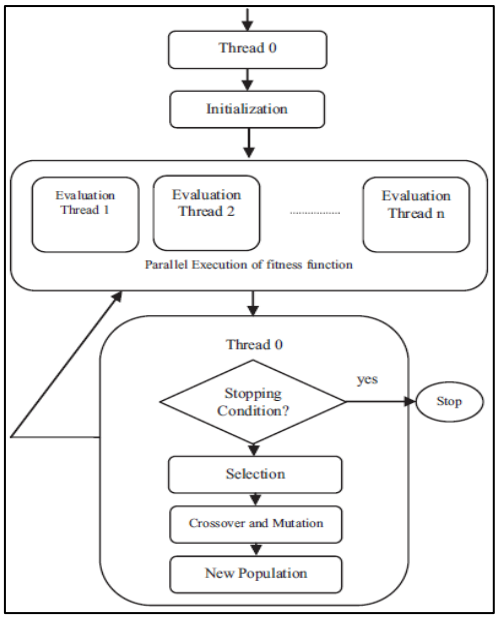

Fig. 2: Master Slave Parallel Genetic Algorithm model

1) Steps at master

Dataset, threshold support and threshold confidence, no. of iterations, Crossover percentage and Mutation Percentage is taken from user.

- Apply Apriori Algorithm.
- Generate the population for applying Genetic Algorithm.
- Split the population in n+ 1 part. (n is no of connected slaves)
- Distribute the population to the number of connected slaves.
- Collect the best rules from all the connected slaves.
- Send the most optimal rules as best rules to the user.

2) Steps at slave

Slave applies the Genetic Algorithm Technique to the rules received from master and sends the results back to the master.

- Receive the Population from master.
- Apply Genetic Algorithm.
- Send these best rules to master.

C. Optimistic rule generation using Hierarchical Genetic Algorithm

Master splits the population, sends it to its connected slaves and applies Genetic Algorithms to its own population [22]. Now the slaves work as master and split the received population again and send it to its connected slaves. The hierarchy is maintained between processors and finally the master gets the output representing best rules. This is shown in Fig.3.

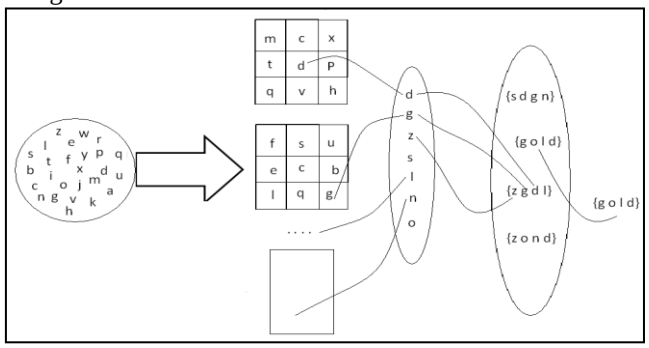

Fig. 3: Optimistic Rule generation using Hierarchical Genetic Algorithm model

It has been emphasized already that our goal is to discover not a single rule but a useful collection of different rules. To achieve this we shall use genetic algorithms operating at two different levels of a hierarchy. The "low-level" genetic algorithms, slave will search for individual rules competitively. (In fact, a fine-grained structured population model will be used for this to encourage a degree of specialisation even at the low level, not least because such structured populations tend to be more effective even for strict optimisation problems.

A higher level genetic algorithm, master, will then take rules generated by the lower level algorithms and use them as basic 'genetic material' from which to form sets of rules using techniques for genetic set-based optimization. More precisely, rules will be taken from each of the low-level populations to form a universal set of rules from which it will be the task of the high level genetic algorithm to find the 'best' set of some given size. In the high level genetic algorithm the fitness function is applied to entire sets of rules and it is these sets that compete.

The aim is to find the best collection of rules with regard to a balance of rules with different characteristics. In this manner, competition at two different levels of the hierarchy results in the discovery 3 of co-operatively useful sets of rules. The low level populations search for good areas of the search space and the high level population combines these into a useful coverage. The rough or low-level genetic algorithms use structured populations to search competitively for individual rules (or more generally, good candidate elements for a set).

The high-level genetic algorithm searches for the best set of rules it can form using those individual rules generated by the low-level genetic algorithms (or, in general, the best set it can form from the elements found by the low-level genetic algorithms). To determine a suitable evaluation function for the high level algorithm we need to consider the characteristics we would like the collections of rules presented to us by a data miner to have. Clearly we would like each rule individually to be of high utility, but perhaps more importantly we wish the rules presented to us to be significantly different from one another, covering as many different rule types as possible.

The performance of a parallel and hierarchical Genetic Algorithm is determined by calculating its speed-up. Speed-up is defined as the ratio of the worst-case execution time of the fastest known sequential Genetic algorithm for a particular problem to the worst-case execution time of the parallel or hierarchical Genetic algorithm. Total cost of a parallel or hierarchical algorithm is the product of time complexity and the number of processors used in that particular algorithm.

IV. EXPERIMENTAL SETUP AND RESULTS

The system is developed in Java and is run in the Net beans environment. The Input used for the development of the system was a sample dataset of a super market. It has 697 rows and 36 valuable attributes, the Genes from 50 available items in the supermarket.

A. Association rule analysis using Genetic Algorithm

For implementing this section of the system following inputs are given - configuration file, Threshold support value and

Threshold confidence value, number of iterations, Crossover rate and Mutation rate. We declare a default configuration, a regular transaction file with an item separator. Configuration File deals with the structure in which nodes are arranged.

B. Association rule analysis using Parallel Genetic Algorithm

For implementing this section of the system, we divide the total set into partition and apply Genetic algorithm to each of the partitions. We Enter configuration file, Threshold Support value, Threshold Confidence value, number of Iterations, Crossover rate and Mutation rate.

C. Association rule analysis using Hierarchical Genetic Algorithm

For implementing this section of the system, we divide the total set into partition and apply Genetic algorithm to each of the partitions arranged in an hierarchical structure. We Enter configuration file, Threshold Support value, Threshold Confidence value, number of Iterations, Crossover rate and Mutation rate.

A default configuration i.e. a regular transaction file with an item separator is declared. Configuration File arranges the nodes in a hierarchical configuration for computation.

Comparison of time taken by different variants of Genetic Algorithms to produce best rules is shown in Table 1.

Genetic Algorithm	Parallel Genetic Algorithm	Hierarchical Genetic Algorithm
35.625seconds	21.344seconds	20.23 seconds

Table 1: Time Required For Different Variants of Genetic Algorithms to Produce Best Rules for the Same Data Set

The table shows that time required for generating rules by Hierarchical Genetic algorithms is the least as compared to Genetic algorithms and Parallel Genetic Algorithms.

The speed-up of a Parallel Genetic Algorithm implementation is calculated as

$$S\_pga = \frac{W\_s}{W\_pga} \qquad (3)$$

Where, S_pga is speed-up of a Parallel Genetic Algorithm, W_s is the worst case execution time of the fastest known sequential for a particular Genetic Algorithm problem and W_pga is the worst case execution time of the parallel Genetic algorithm.

We have gained a speed-up of 1.67 for parallel Genetic Algorithm implementation and for hierarchical Genetic Algorithm implementation speed-up is,

$$S\_hga = \frac{W\_s}{W\_hga} \qquad (4)$$

Where, S_hga is the speed-up of Hierarchical genetic algorithms, W_s is the worst case execution time of the fastest known sequential for a particular Genetic Algorithm problem and W_hga is the worst case execution time of the hierarchical Genetic algorithm. Therefore speed-up for hierarchical Genetic Algorithm achieved by us is 1.76.

V. Conclusion

We have dealt with a challenging problem of finding optimized association rules using Genetic Algorithms and its variants Parallel Genetic Algorithms and Hierarchical Genetic Algorithms. The frequent item-sets are generated using the Apriori association rule mining algorithm. We have proposed different variant structures to apply Genetic Algorithms for finding optimized rules.

As per the results generated experimentally, Optimistic Rules generation using Hierarchical Genetic Algorithm will give a speedup of 1.76 and hence take less time for result generated as compared to Parallel Genetic Algorithms having a speedup of 1.67 as compared to Simple Genetic Algorithms in its primitive form.

Acknowledgment

The authors would like to thank their affiliated institutions for all the support given to them during their research and preparing of this publication.

References

[1] M.Zhu and H. A. Chipman, "Darwinian evolution in parallel universes: a parallel genetic algorithm for variable selection," in Technometrics, vol. 48, no. 4, 2006, pp. 491–502.
[2] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in Proceedings of the 1993 ACM Sigmod International Conference on Management of Data, Washington DC (USA), 1993, pp. 207–216.
[3] K. F. Man, K. S. Tang, and S. Kwong, "Genetic algorithms: Concepts and applications," in IEEE Transactions on Industrial Electronics, vol. 43, no. 5, 1996, pp. 519–534.
[4] D. Weile and E. Michielssen, "Genetic algorithm optimization applied to electro-magnetic: a review," in IEEE Transactions on Antennas and Propagation, vol. 45, no. 3. IEEE, 1997, pp. 343–353.
[5] K. Tang, K. Man, S. Kwong, and Q. He, "Genetic algorithms and their applications" in IEEE Signal Processing Magazine, Vol.45, No. 3, March 1997. IEEE, 1996, pp. 22–37.
[6] C. N. Giap and D. T. Ha, "Parallel genetic algorithm for minimum dominating set problem," in proceedings of International Conference on Computing, Management and Telecommunications (ComManTel), added to IEEE Xplor, 2014.
[7] J. J. Grefenstette and J. E. Baker, "How genetic algorithms work: A critical look at implicit parallelism," in Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann Publishers Inc., 1989, pp. 20–27.
[8] A. Bienz, K. Fokle, Z. Keller, E. Zulkoski, and S. Thede, "A generalized parallel genetic algorithm in erlang," in Proceedings of the Mid states Conference on Undergraduate Research in Computer Science and Mathematics, 2011.
[9] M. Nowostawski and R. Poli, "Parallel genetic algorithm taxonomy," in Proceedings of the KES99, 1999, pp. 1–5.
[10] R. Murphy, "A generic parallel genetic algorithm," in A Thesis submitted to The University of Dublin for the degree of M.Sc. in High Performance Computing Department of Mathematics University of Dublin Trinity College, 2003, pp. 1–68.

[11] R. C. Correa, A. Ferreira, and P. Rebreyend, "Scheduling multiprocessor tasks with genetic algorithms," in IEEE Trans. Parallel Distrib. Syst, vol. 10, no. 8. IEEE Press, 1999, pp. 825–837.
[12] M. Golub and S. KASAPOVIC, "Kasapovic scheduling multiprocessor tasks with genetic algorithms" in Proceedings of Applied Informatics Proceedings, 2002.
[13] R.Lisboa, E. Yasojima, M. Rodrigo, S. de Oliveira, and M. F. Mollinetti, "Parallel genetic algorithm with social interaction for solving constrained global optimization problems" in IEEE Xplore, 2016.
[14] P. Francq, "Hierarchical genetic algorithms," in O Paul Otl et.at. http://www.otletinstitute.org/wikics/ Hierarchical-Genetic-Algorithms, 2013.
[15] A. Paszynska and M. Paszynski, "Application of a hierarchical chromosome based genetic algorithm to the problem of finding optimal initial meshes for the self-adaptive hp-fem," in Computing and Informatics, vol. 28, no. 5, 2009, pp. 1001–1015.
[16] M. Gulsen and A. E. Smith1, "A hierarchical genetic algorithm for system identification and curve fitting with a supercomputer implementation," in IMA Volumes in Mathematics and its Applications - Issue on Evolutionary Computation (Springer-Verlag), 1998, pp. 1–33.
[17] L. Yu, Z. Shen, C. Miao, and V. Lesser, "Genetic algorithm aided optimization of hierarchical multiagent system organization," in Computer Science Technical Report UM-CS-2011-003, University of Massachusetts at Amherst, 2011, pp. 1169–1170.
[18] A. K. Korkut Koray GUNDOGAN, Bilal ALATAS, "Mining classification rules by using genetic algorithms with non-random initial population and uniform operator," in Turk J Elec Engin, vol. 12, no. 1. TUBITAK, 2004, pp. 43–52.
[19] S. Ghosh, S. Biswas, D. Sarkar, and P. P. Sarkar, "Mining frequent item-sets using genetic algorithm," in International Journal of Artificial Intelligence Applications, vol. 1, no. 4, 2010, pp. 133–143.
[20] D. L. A. de Araujo, H. S. Lopes, and A. A. Freitas, "A parallel genetic algorithm for rule discovery in large databases," in Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, 1999.
[21] S. R. Dash, S. Dehuri, and S. Rayaguru, "Discovering interesting rules from biological data using parallel genetic algorithm," in 3rd IEEE International Advance Computing Conference, 2013, pp. 631–636.
[22] N. J. Radcliffe and P. D. Surry, "Co-operation through hierarchical competition in genetic data mining," in proceedings of Edinburgh Parallel Computing Centre - TR94-09, 1994, pp. 1–10.

