Discovering Interesting Rules from Biological Data Using Parallel Genetic Algorithm

Satya Ranjan Dash
School of Computer Application
KIIT University
Patia ,Bhubaneswar,
Odisha, India-751024
E-mail: sdashfca@kiit.ac.in

Satchidananda Dehuri
Department of Systems Engineering,
Ajou University, San 5, Woncheon-dong,Yeongtong-gu,
Suwon 443-749, Republic of Korea.
E-mail: satchi@ajou.ac.kr

Susil Rayaguru
School of Computer Application
KIIT University
Patia ,Bhubaneswar,
Odisha, India-751024
E-mail: susil@rayaguru.com

Abstract— **In this paper, a parallel genetic based association rule mining method is proposed to discover interesting rules from a large biological database. Apriori algorithms and its variants for association rule mining rely on two user specified threshold parameters such as minimum support and minimum confidence which is obviously an issue to be resolved. In addition, there are other issues like large search space and local optimality attracts many researchers to use heuristic mechanism. In the presence of large biological databases and with an aim to circumvent these problems, genetic algorithm may be taken as a suitable tool, but its computational cost is the main bottle-neck. Therefore, we choose parallel genetic algorithms to get relief from the pain of computational cost. The experimental result is promising and encouraging to do further research especially in the domain of biological science.**

Keywords— Apriori algorithm, Genetic algorithm, Parallel genetic algorithms, Association rule mining, Data mining

I. INTROUDUCTION

A parallel genetic based association rule mining method is proposed to discover interesting rules from a large biological database or biomedical dataset. Association rule mining depends upon two user specified thresh hold value known as support and confidence. Apriori algorithms for association rule mining also rely on two user specified threshold parameters such as minimum support and minimum confidence. However, there are certain challenges in applying apriori like algorithm, e.g., database dependent minimum support and large search space. Hence, in the presence of large biological databases, it is a difficult task to guess the threshold value for minimum support.

To avoid these problems, genetic algorithm may be considered as a suitable tool, but its computational cost is the main bottle-neck. Therefore, we choose parallel genetic algorithms to get relief from the pain of computational cost. In our work, it is not required to give the user specified minimum support or minimum confidence value, it gets automatically generated through the genetic algorithm. As sometimes, if we provide it by the user, we may find some interesting patterns miss out which is having less value as minimum support or minimum confidence.

II. PRELIMINARIES

Database-dependent minimum-support means that users must specify suitable thresholds for their mining tasks though they may have no knowledge concerning their databases. To avoid these problems, in this paper, we intend to use an evolutionary mining strategy in which association rule mining based on a genetic algorithm has been implemented. It has been observed that the fitness evaluation in genetic algorithm is mostly the expensive step; hence to minimize the overall computational complexity of genetic algorithm it is indeed to compute the fitness in parallel. A model is illustrated in Fig. 2.

A. Association Rule Mining

Association rule mining is one of the most important rules of data mining, used to extract interesting correlations, frequent patterns, and associations among a set of items in the transaction database. Due to its high degree of implementation in areas such as telecom networks, risk management, inventory control, etc., association rule mining has been one of the well researched techniques of data mining.

Before we dive into details of association rule mining we shall define some basic terminology. If we think of the universe as the set of items available at the store, then each item has a boolean variable representing the presence or absence of that item. So each collections of items called as an itemset, can be represented by a boolean vector of values assigned to these variables. These vectors can be analyzed for buying patterns that select items that are frequently associated or purchased together. These patterns can be represented in form of association rules. Hence an association rule is about relationships between two disjoint itemsets X and Y. The statement $X \Rightarrow Y$ implies the pattern when X occurs, Y occurs.

Rule support and confidence are two measures of how interesting a rule is. A support of 2% says that out of all transactions, 2% show that X and Y are bought together. Whereas, a confidence of 70% says that 70% of customers who purchased X also bought Y. Let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of items. Let D be a database where each transaction T is a set of items such that $T \subseteq I$. The rule $X \Rightarrow Y$ has a support s and confidence c, where s is the probability of transactions in D

978-1-4673-4529-3/12/$31.00 ©2012 IEEE

containing X⋃Y and c is the probability of transactions in D containing X that also contain Y. A detailed analysis can be found in [1].

B. Parallel Genetic Algorithms

Genetic algorithm (see Fig-1) is a heuristic search algorithm that is inspired by the evolutionary process of natural development. This heuristic is normally used to make useful solutions to optimization and search problems. It belongs to well-built class of evolutionary algorithms (EA).

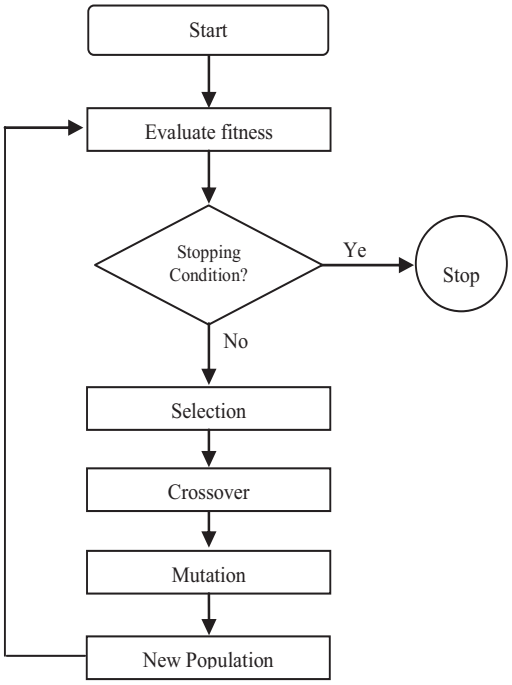

Fig. 1: Flow graph of GA

Canonical GA runs in an iterative method by generating new populations of strings from the old ones. Every string is the encoded (binary, real, etc.) version of a tentative solution. An assessment function associates a fitness measure to every string indicating its suitability to the problem. The algorithm applies stochastic operators such as selection, crossover, and mutation on an initial random population in order to compute a whole generation of new strings [2].

Parallel GAs (PGAs) in Fig-2 is not just parallel version of canonical genetic algorithms. In fact they reach ideal goal of having a parallel algorithm whose presentation is better than the sum of the separate behaviours of its component sub-algorithms, and this is why we directly focus on them [3].

A large population distributed among a number of semi-isolated breeding groups is known as polytypic. A PGA introduces the model of interconnected demes. The local selection and reproduction rules allow the species to grow locally, and diversity is superior by migration of strings among demes [4].

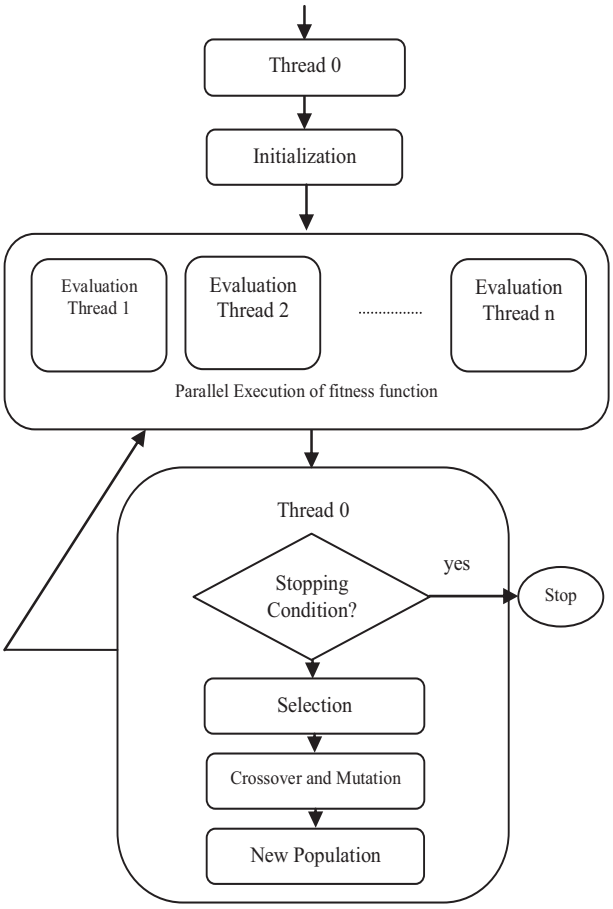

Fig. 2: Flow graph of pGA

III. RELATED WORK

PGAs are parallel stochastic algorithms. Like sequential genetic algorithms (GAs) [5][6], they are based on the natural evolutionary standard. Better individuals survive and reproduce themselves more often than the worse ones. To speed up the processing of generations of populations, we can split the population into several sub-populations and run them in the parallel way.

Very important theoretical questions were raised about *comparison of quality* solutions between a PGA and a classic GA [7]. They claimed that relatively remote demes converge to dissimilar solutions and that relocation and recombination combine partial solutions.

The population mass is also the main thing to decide the time that the GA desires to find the solution. Therefore, the deme sizing models may be used to predict the implementation time of the parallel GA, and to match up to it with the time required

by a serial GA to reach a solution of the same quality. Cantú[8] integrated the deme sizing models with a model for the communications time, and predicted the expected parallel speed-ups for the two bounding cases.

This section summarizes some new advancement in the theoretical study of parallel GAs. An important observation on master-slave GAs is that as additional processors are used, the time to evaluate the fitness of the population decreases. But at the same time, the cost of transfer the individuals to the slaves increase. This tradeoff between diminishing computation times and increasing communication times entails that there is an optimal number of slaves that minimizes the total execution time.

The idealized bounding models can be complete in several instructions, for instance to judge lesser migration rates or more lightly connected topologies. The information that there is an finest number of demes restrictions the processors that can be used to decrease the finishing time. Using more than the optimal number of demes is wasteful, and would result in a slower algorithm. Hierarchical parallel GAs can use more processors effectively and reduce the execution time more than a pure multiple-deme GA.

IV. PROPOSED WORK

As we know, the user specified parameters minimum support and minimum confidence plays a crucial role in Association Rule Mining for finding interesting rules. But it becomes a major challenge for the user to specify the parameter as minimum support and minimum confidence when the data base contains overwhelming number of records. When the database is large that becomes disk resident, this requires reading the database completely for each pass, resulting in large number of disk I/Os which ultimately decreases the productive of the algorithm.

In this paper we have employed PGA with association rule mining, where the interestingness of the rules is governed by the fitness function, hence the user is never required to specify the initials parameter as minimum support and minimum confidence.

In the fitness function we are evaluating the positive confidence of each the frequent item set to find the interestingness. Hence the evaluation period of the fitness function is proportionately related with the number of records present in the database. Therefore to optimize the evaluation of the fitness function we have employed a parallel model for the fitness function evaluation which is not inherently sequential.

B. Encoding for GA

Further, we are adhered the Michigan strategy of encoding the chromosome where each association rule is represented as a single chromosome.

The items is being represented by the set I as $I = \{i_1, i_2, i_3, \ldots, i_n\}$ with their indexes. Give an association *K* rule $X \rightarrow Y$ is represented as a single chromosome where *X* and Y is represented as the subset of I. And $X \bigcap Y = \Phi$, where X is called the antecedent and *Y* is called as the consequent. The chromosome is depicted in the Fig-3. Here X and Y can be specified as a frequent item set. In the proposed algorithm we are finding all possible frequent item set of $X \rightarrow Y$ for association rule mining. Subset *X* and *Y* are separated in a chromosome by the index m which will be chosen randomly such as $X=C_1,C_2,\ldots,C_m$ and $Y= C_{m+1},\ldots,C_k$. $0<m<k$

C_1	C_2	$C_{\ldots}$	C_m	C_{m+1}	C_{k-1}	C_k

Fig.3: Representation of the *k* rule chromosome

B. Operators for GA

For the implementation of genetic algorithm, we have employed three Genetic Operators such as selection, crossover, and mutation.

Function *select(Chromosome c)* evaluates the chromosome according to the fitness criteria. It returns TRUE if the chromosome qualify the fitness test else returns FALSE

```
Boolean select(Chromosome c)
begin
        if(fitness(c) >= 0)then
                return TRUE;

        else
                return FALSE;
end
```

Function mutate(c, pm) performs a mutation occasionally in the chromosome. The function random (k+1) returns a random integer between 0 to k. Function rand () return a floating value between 0 and 1.

```
Chromosome mutate (Chromosome c, pm)
begin
        if(rand() < pm) then
        begin
                rand1 ← random(1,k+1);
                rand2 ← random(1,n+1);
                c[rand1] ← I[rand2];
        end
        return c
 end
```

Function *crossover(population)* is used for generating new offspring using the current population. We have employed two point strategies to crossover two chromosomes which give birth to new offspring.

```
Crossover (population, pc)
begin
    populationTemp ← Ø;
    for ∀ Ci = (Ci1,Ci2,..,Cik)  population do
    begin
        for ∀ Cj = (Cj1,Cj2,..,Cjk)  population and  Ci ≠ Cj do
        begin
            if ( i ≠ j) then p ← random(k+1)
                q ← random(k+1)
                q ← max(p,q)
                p ← min(p,q)
            C3 ← (Ci1,Ci2,..,Cip,Cjp+1,…Cjq,Ciq+1,Cik)
            C4 ← (Cj1,Cj2,..,Cjp,Cip+1,…Ciq,Cjq+1,Cjk)
            populationTemp  C3, C4
            end of if
            end of for
            end of while
return populationTemp;
end
```

C. Fitness Evaluation

Fitness function plays a crucial role in determining the interesting association rule. It employs fitness criteria for filtering out the interesting chromosome.

In the aforesaid algorithm we have defined the fitness function as

fitness (c) =

$$\frac{supp(C1,C2,\dots,Ck) - supp(C1,C2,\dots,Cm)supp(Cm+1,\dots Ck)}{supp(C1,C2,..,Cm)(1-supp(Cm+1,\dots,Ck)}$$

We have employed parallel computation for the above fitness function for evaluation for optimizing the computation time.

D. *Initial Population*

For GA implementation we need the initial population, which can be derived by applying repetitive mutation *mutate (Chromosome s, pm)* operation over a single seed chromosome.

In the above function *MAXPOPULATION* is the user specified constant that denotes the maximum number of chromosomes that can exist in the initial population.

```
population initialize(Chromosome c)
  begin
      Initialpop[0] ← c;
      while Initialpop [0] < MAXPOPULATION / 2 do
       begin
        populationTemp ← Ø;
          for ∀ c ∈ Initialpop[0] do
       begin
   populationTemp ←  populationTemp ⋃ mutate(c,1);
        end
        Intitalpop[0] ←  Intitalpop[0] ⋃ populationTemp
       end
      return Initialpop [0]
  end
```

E. Main Algorithm

The current population is represented as pop[i]. Here selection is applied for retaining interesting chromosomes in the population which produce a new population as pop[i+1]. Any pair of chromosomes in the new population is crossed over to produce two new offspring. This algorithm terminates with a population which contains high quality chromosomes.

```
population main(s,ps,p c,pm)
begin
      i ← 0;
      pop[i] ← initialize(c)
while not terminate(pop[i]) do
      begin
          pop[i+1] ← Φ ;
          popTemp ← Φ  ;
          for ∀ c ∈ pop[i] do
            if select(c,ps) then
              pop[i+1] ← pop[i+1] ⋃ c;
          end
          popTemp ← crossover(pop[i+1],pc);
        for ∀ c ∈ popTemp do
          pop[i+1] ← (pop[i+1] – c) ⋃ mutate(c,pm);
          i ← i+1;
      end
      return pop[i];
end
```

Termination criteria for the aforesaid algorithm

1. The difference between the best and the worst chromosome is less than a given value $\propto$.
2. The number of iterations exceeds a given maximum number *maxloop*.

V. EXPERIMENTAL STUDY AND RESULTS

We have employed the specified algorithm in the Post-Operative Patient Data Set from UCI. The data set contains 90 records and 7 attributes.

A. *Data Set Information*

The classification task of this database is to determine where patients in a postoperative recovery area should be sent to next. Because hypothermia is a significant concern after surgery, the attributes correspond roughly to body temperature measurements.

Attribute Information:

- L-CORE (patient's internal temperature in C): high (> 37), mid (>= 36 and <= 37), low (< 36)
- L-SURF (patient's surface temperature in C): high (> 36.5), mid (>= 36.5 and <= 35), low (< 35)
- L-O2 (oxygen saturation in %): excellent (>= 98), good (>= 90 and < 98), fair (>= 80 and < 90), poor (< 80)
- L-BP (last measurement of blood pressure): high (> 130/90), mid (<= 130/90 and >= 90/70), low (< 90/70)
- SURF-STBL (stability of patient's surface temperature):
  stable, mod-stable, unstable
- CORE-STBL (stability of patient's core temperature) stable, mod-stable, unstable
- BP-STBL (stability of patient's blood pressure) stable, mod-stable, unstable

I (patient sent to Intensive Care Unit),
S (patient prepared to go home),
A (patient sent to general hospital floor)

The specified algorithm was exhaustively tested with the following parameter which shown in Fig. 3

maxloop 50~5000 and α value set as 0.03, ps 0.92, pc as 0.80, pm 0.02, k as 3, population size as 100 and we achieved more than 85% of accuracy.

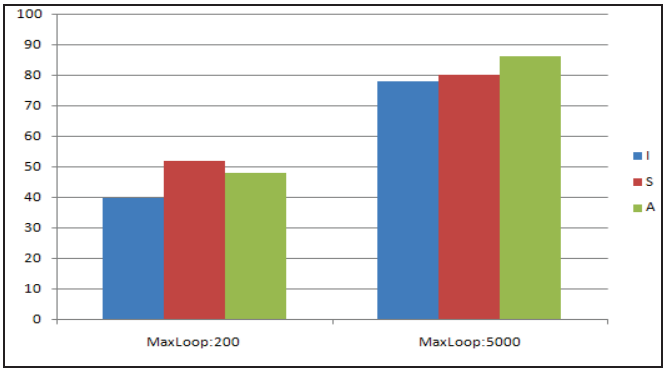

Fig.4. Accuracy Graph

VI. CONCLUSION AND FUTURE WORK

It becomes a major challenge for the user to specify the said parameter when the data base is overwhelmingly large. In this paper, we have proposed PGA based association rule mining, where the interestingness of the rules is governed by the fitness function where the user specified parameter could be eliminated.

This work can further be extended to create a hybrid model using soft set and PGA to overcome incomplete and noisy data for finding the interesting rules.

REFERENCES

[1] J. Han, M. Kamber, Data Mining Concepts and Techniques, Morgan Kaufmann Publishers, ISBN 978-81-312-0535-8 2010.

[2] T.Bäck, D. Fogel, Z. Michalewicz (eds.) *Handbook of Evolutionary Computation*. Oxford University Press. 1997.

[3] E. Cantú-Paz. "A Summary of Research on Parallel Genetic Algorithms". *R. 95007*, July 1995. revised version, *IlliGAL R. 97003*. May 1997.

[4] A. Chipperfield, P. Fleming. "Parallel Genetic Algorithms". *Parallel and Distributed Computing Handbook*, A. Y. H. Zomaya (ed.), MacGraw-Hill, pp. 1118-1143. 1996.

[5] A. Grajdeanu. *Parallel Models for Evolutionary Algorithms*. ECLab, George Mason University, 38, 2003.

[6] J. J. Grefenstette. *Parallel adaptive algorithms for function optimization*. Report No. CS-81-19, Vanderbilt University,TN, 1981.

[7] T. Starkweather, D. Whitley, K. Mathias. Optimization using distributed genetic algorithms. *Parallel Problem Solving from Nature*, Berlin, Germany, 176-185, 1991.

[8] E. Cantu -Paze, D. E. Goldberg, "Modeling Idealized Bounding Cases ofParallel Genetic Algorithms", *Proceedings of the Second Annual Conference*, Morgan Kaufmann (San Francisco, CA), 1997.

[9] S. Yan, C. Zhang ,S. Zhang - ARMGA: Idetifying Interesting Association Rules with Genetic Algorithm Applied Artificial Intelligence, 19:677–689 , 2005

[10] B. Shumeet, "The evolution of genetic algorithms: Towards massive parallelism," in *Proceedings of the Tenth International Conference on Machine Learning*, San Mateo, CA,, Morgan , pp. 1–8, 1993

[11] E. Cant´u-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers, 162, 2000.

[12] H. Mühlenbein. "Evolution in Time and Space - The Parallel Genetic Algorithm". *Foundations of Genetic Algorithms*, G. J. E. Rawlins (ed.), Morgan Kaufmann, pp. 316-337. 1991.

[13] D. E. Goldberg. "Sizing Populations for Serial and Parallel Genetic Algorithms". *Proceedings of the 3rd ICGA*, J. D. Schaffer (ed.), Morgan Kaufmann, pp. 70-79. 1989.