Parallel Island Model for Attribute Reduction

Mohammad M. Rahman[1], Dominik Ślęzak[1,2], and Jakub Wróblewski[2]

[1] Department of Computer Science, University of Regina
Regina, SK, S4S 0A2 Canada
{rahman5m,slezak}@cs.uregina.ca
[2] Polish-Japanese Institute of Information Technology
Koszykowa 86, 02-008 Warsaw, Poland
{slezak,jakubw}@pjwstk.edu.pl

Abstract. We develop a framework for parallel computation of the optimal rough set decision reducts from data. We adapt the island model for evolutionary computing. The idea is to optimize reducts within separate populations (islands) and enable the best reducts-chromosomes to migrate among islands. Experiments show that the proposed method speeds up calculations and also provides often better quality of results, comparing to genetic algorithms applied so far to the attribute reduction.

1 Introduction

Feature Selection in KDD and Pattern Recognition is an essential task [6]. Rough set-based methods can measure multi-attribute relationships. Hence, they are used to identify (ir)relevant features. Selection of features that (approximately) preserve so called indiscernibility of objects leads to classifiers based on reducts – most informative, irreducible subsets of attributes [11].

There can be many reducts in a data set. Finding short reducts is a major task while developing a good rough set-based classifier. It enables to overcome weakness of other approaches that ignore the effects of the feature subsets on performance of the induction algorithms. The problem of finding minimal reducts is NP-hard [11]. Some heuristic approaches were developed. An order-based genetic algorithm (o-GA) for finding minimal reducts [14] is proposed as a hybrid process. Genetic algorithm (GA) greedily selects features and rough set methods measure a degree of their relevancy within a given subset.

o-GA for finding short reducts works on a single processor. Its weakness is that it takes more time to find out sufficiently good reducts in a large search space. We use distributed evolutionary computing [2] to exploit availability of computer networks and massive power of parallel computing. In distributed environment, the total population is divided into sub-populations evolving in parallel, which increases performance of calculations. But it declines the overall average quality of minimal reducts and sometimes it can not avoid local optima. It turns out that the proposed island model implementation in a distributed environment, which exploit migration technique to exchange genetic material between populations, increases quality along with performance.

The paper is organized as follows: Section 2 introduces basics of the rough set theory. Section 3 introduces order-based genetic algorithm for reduct generation. Section 4 gives an idea of the island model of distributed computing. Section 5 summarizes results comparing to the single processor-based rough-genetic approaches. Section 6 gives conclusions and discusses future work.

2 Rough Sets

Rough sets were introduced by Zdzisław Pawlak in 1982. It has become a popular theory in the field of data mining, derived from fundamental research on logical properties of information systems – pairs $A = (U, A)$, where U is a non-empty finite set called the universe and A is a non-empty finite set of attributes, i.e. $a : U \rightarrow V_a$ for $a \in A$, where V_a is called the value set of a.

With any $B \subseteq A$, there is associated the equivalence relation $IND(B) = \{(x, x') \in U^2 \mid \forall a \in B, a(x) = a(x')\}$, called the B-indiscernibility relation. If $(x, x') \in IND(B)$ then objects x and x' are indiscernible from each other by attributes from B. Equivalence classes of $IND(B)$ are denoted by $[x]_B$.

A decision system takes the form of $A = (U, A \cup \{d\})$ where d is the decision attribute. Elements of U are called objects. For every value $v_k \in V_d$ we define the k-th decision class $X_k = \{u \in U : d(u) = v_k\}$. For every $X_k \subseteq U$ and attribute subset $B \subseteq A$, we can approximate X_k by the B-lower approximation $\underline{B}X_k$ and B-upper approximation $\overline{B}X_k$ using knowledge of B. $\underline{B}X_k$ is the set of objects that are surely in X_k, defined as $\underline{B}X_k = \{x | [x]_B \subseteq X_k\}$. $\overline{B}X_k$ is the set of objects that are possibly in X_k, defined as $\overline{B}X_k = \{x | [x]_B \cap X_k \neq \emptyset\}$.

The positive region of d with respect to condition attributes B is denoted by $POS_B(d) = \bigcup \underline{B}X_k$. It is a set of objects of U that can be classified with certainty employing attributes of B. A subset $R \subseteq B$ is said to be a reduct of B if $POS_R(d) = POS_B(d)$ and there is no $R' \subsetneq R$ such that $POS_{R'}(d) = POS_R(d)$. In other words, a reduct is the irreducible set of attributes preserving the positive region. There can be many such reducts in a decision system.

3 Order-Based Genetic Algorithm

Genetic algorithm (GA) is an adaptive heuristic search method for solving optimization problems. It was introduced by John Holland in 1970s. As an alternative technique, it outperforms most of traditional methods. Finding the minimal reducts is a NP-hard problem [11]. Hence, GA is a good candidate as a methodology for finding minimal reducts.

In classical GA, individuals are encoded as binary strings of the attributes (e.g. $0100110100 \equiv \{a_2, a_5, a_6, a_8\}$). Each individual represents a set of attributes generated by mutation, crossover and selection procedures using some fitness criteria. Individuals with maximal fitness are highly probable to be reducts but there is no full guarantee. A hybrid approach using order-based encoding of the attributes is proposed in [14] where each individual will produce a reduct.

Algorithm 1 Reduct Calculation from a chromosome

Input: chromosome $\tau \equiv (a_1, a_2, a_3, ..., a_n)$
Output: reduct

1: $R = \{a_1, ..., a_n\}$
2: **for** $i = 0$ to n **do**
3: **if** $POS_{R-\{a_{n-i}\}}(d) = POS_R(d)$ **then**
4: $R = R - \{a_{n-i}\}$
5: **end if**
6: **end for**
7: return R

Here, an individual is an ordered list of features $(a_1, a_2, a_3, ..., a_n)$. Order is a permutation of the features generated by GA operators. A deterministic algorithm is used to calculate the reduct (denoted by R) from an individual. Crossover, mutation and selection are applied to generate the next generation population. For selection method, length (denoted by L_R) of R is used as fitness. Deterministic procedure keeps removing each feature from the list's end as long as the remaining features maintain the same positive region.

4 Parallel GA and Island Model

Parallel GA was first attempted by Grefenstette in [5]. Parallelism refers to many processors, with distributed operational load. Each GA is a good candidate for parallelization. Processor may independently work with different parts of a search space and evolve new generations in parallel. This helps to find out the optimum solution for the complex problems by searching massive populations and increases quality of the solutions by overcoming premature convergence. Many complex problems like: set partitioning problem [8], RNA folding pathways [10], multiprocessor scheduling problem [2] are treated with Parallel GA.

The above traditional parallel GA is called a sequential GA. Another type is called the Island Model (IM) [12], where processors are globally controlled by message passing within master-slave architecture. Master processor sends "START" signal to the slave processors to start generations and continue sending "MIGRATION" message to partially exchange the best chromosomes between the processors. Time between two consecutive "MIGRATION" signals is called the migration step; percentage of the best chromosomes is called migration percentage. The worst chromosomes are replaced by the received ones. Migrations should occur after a time period long enough for allowing development of good characteristics in each sub-population.

5 Experimental Results

We calculate reducts within Cygwin-based (a Linux-like environment in Windows) parallel processing framework, using modified libGA [3]. We tested:

Algorithm 2 Island Model (master pc)

Input: number of processors N
Output: reducts

```
for i = 0 to N do
   SendMessage(i,START)
end for
while short enough reducts not found do
   for i = 0 to N do
      SendMessage(i,NEXTGENERATION)
   end for
   CollectReducts()
   if current generation such that migration time reached then
      for i = 0 to N do
         SendMessage(i,MIGRATION)
      end for
   end if
end while
```

1. Order-based serial GA (OGA) using a single processor
2. Parallel island model (PIM) over 10 computers

Each computer was Pentium 997 MHz, 384 Mb of RAM, using Windows 2000. We considered four UCI data sets. Partially matched crossover (PMX) with the rate of 70% and swap mutation, rate 1%, were used. We took a pool size 300 for OGA. We split it among 10 processors (30 chromosomes for each) for PIM.

PIM was done by two methods: with and without migration. In PIM with migration (further denoted by PIM-Mg), two new parameters are added: ms (migration step) is the number of generations between migrations; mr (migration rate) is the percentage of individuals migrating to the neighbor processors. In our experiment, we used $ms = 5$ and $mr = 25\%$.

For OGA, evolution was continued until the chance of getting new reducts was too low. Then we tried to find the same number of minimal reducts (only reducts of minimal length L and $L + 1$) by PIM methods. The process was repeated 20 times, given random initial pool. Average results are presented below.

Table 1. Average time taken in three different methods for four data sets, with different number of objects (m) and attributes (n). The top plus signs in the three results for PIM indicate that the minimal reducts could not be found within 500 generations.

m	n	No. of Min. Reducts	OGA	PIM	PIM-Mg
70	18	142	49.75	53.45^{+}	12.06
103	19	166	77.6	19.40	8.70
307	36	529	4980	1803^{+}	1289
592	19	1434	11219	2460^{+}	1778

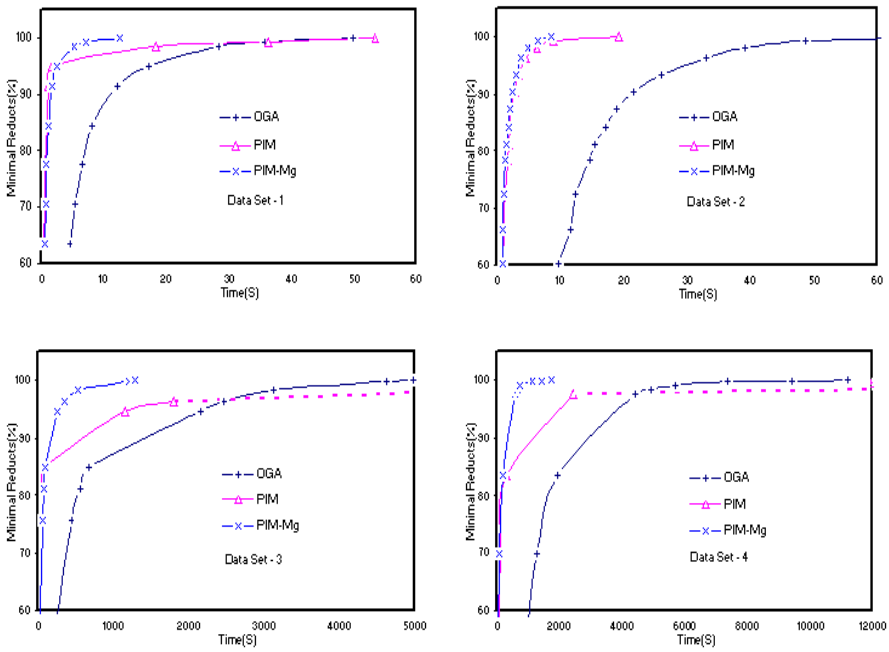

Fig. 1. The minimal reduct generation process for four different data sets, displayed in terms of "minimal reducts-time" graphs. The dotted lines in data sets 3 and 4 indicate that after some generations PIM without migration fell in local-optima. We considered the maximum time as the time needed for finding all the minimal reducts.

6 Conclusions

Results show that PIM takes less time in the initial stage because of not spending any time for migration and total population are distributed in 10 processors. But usually it got trapped in local optima because of its small population size. On the other hand, PIM-Mg outperforms all other methods in time and quality. Moreover, performance increased dramatically for larger data sets.

In conclusion, for feature selection/reduction problems with large search space, we can distribute the total population into islands (processors) to gain performance and apply migration to preserve quality.

In future, we will apply the island model strategy for finding high quality approximate and dynamic reducts. Especially the idea of dynamic reducts – subsets of features remaining reducts for different sub-samples of data [1] – fits the island model perfectly. Assigning different sub-samples to different islands, migrations may provide high stability of best found reducts. Also, we plan to check another possibilities of crossover operators, e.g. OX (order crossover).

Acknowledgments: The research reported in this article was supported in part by research grants from Natural Sciences and Engineering Research Council of Canada awarded to the second author, as well as from Research Centre of PJIIT awarded to the third author.

References

1. Bazan, J., Skowron, A., Synak, P.: Dynamic reducts as a tool for extracting laws from decision tables. In: Proc the Symp. on Methodologies for Intelligent Systems, Charlotte, NC (1994) 16–19
2. Corcoran, A.L., Wainwright, R.L.: A parallel island modal genetic algorithm for the multiprocessor scheduling problem. In: Proc ACM/SIGAPP Symposium on Applied Computing (1994) 483–487
3. Corcoran, A.L., Wainwright, R.L.: LibGA: A user-friendly workbench for order-based genetic algorithm research. In: Proc ACM/SIGAPP Symposium on Applied Computing (1993) 111–118
4. Goldberg, D.E.: Genetic algorithms in search, optimisation and machine learning. Addison-Wesley, Reading MA (1989)
5. Grefenstette, J.J.: Parallel adaptive algorithms for function optimization. Technical Report CS-81-19, Computer Science Department, Vanderbilt University, Nashville, TN (1981)
6. Kittler, J.: Feature selection and extraction. In: Young and Fu (Eds.), Handbook of pattern recognition and image processing. Academic Press, New York (1986) 203–217
7. Knight, L., Wainwright, R.: HYPERGEN – A distributed genetic algorithm on a hypercube. In: Proc the Scalable High Performance Computing Conference. Williamsburg, Virginia (1992)
8. Levine, D.: A parallel genetic algorithm for the set partitioning problem. PhD thesis, Illinois Institute of Technology, Department of Computer Science (1994)
9. Pawlak, Z.: Rough sets – Theoretical aspects of reasoning about data. Kluwer Academic Publishers (1991)
10. Shapiro, B.A., Wu, J.C., Bengali, D., and Potts, M.J.: The massively parallel genetic algorithm for RNA folding: MIMD implementation and population variation. Bioinformatics 17 (2001) 137–148
11. Skowron, A., Rauszer, C.: The discernibility matrices and functions in information systems. In: Słowiński (Ed.): Intelligent Decision Support. Handbook of Applications and Advances of the Rough Sets Theory. Kluwer Academic Publishers, Dordrecht (1992) 331–362
12. Whitley, D.: A genetic algorithm tutorial. Technical report, Colorado State University (1993)
13. Wróblewski, J.: A parallel algorithm for knowledge discovery system, in: Proc PARELEC'98, Bialystok, Poland (1998) 228–230
14. Wróblewski, J.: Finding minimal reducts using genetic algorithms. In: Proc the Second Annual Joint Conference on Information Sciences. September 28 - October 1, Wrightsville Beach, NC (1995) 186–189
15. Wróblewski, J.: Theoretical foundations of order-based genetic algorithms. Fundamenta Informaticae 28(3-4) (1996) 423–430