Pipelined Processing of Genetic Clustering

Malay K. Pakhira
Department of Computer Science and Technology, Kalyani Government Engineering College, Kalyani, WB, India
mkp@kucse.wb.nic.in

Rajat K. De
Machine Intelligence Unit, Indian Statistical Institute, Kolkata, WB, India
rajat@isical.ac.in

Sanghamitra Bandyopadhyay
Machine Intelligence Unit, Indian Statistical Institute, Kolkata, WB, India
sanghami@isical.ac.in

Ujjwal Maulik
Department of Computer Science and Engineering, Jadavpur University, Kolkata, WB, India
ujjwal_maulik@yahoo.com

Abstract

In this article, an attempt is made to solve the clustering problem using a pipelined genetic algorithm, which is a faster version of the conventional genetic algorithm. In the pipelined Genetic Algorithm, the genetic operators are pipelined in such a way that their execution overlaps in time; thereby providing an overall speedup of the process. Since the conventional selection requires the computation of the fitness values of all the chromosomes before the selection process begins, it has been modified in pipelined GA to a version that uses a statistical distribution function. Both crisp and fuzzy clustering problems can be solved using pipelined GA. An analysis of the speedup obtained in case of software pipeline is provided.

Keywords

Clustering, Optimization, Pipelined genetic algorithm, Stochastic selection.

INTRODUCTION

In clustering [1] a set of objects (patterns), generally multi-dimensional in nature, are classified into groups (classes or clusters) such that members of one group are similar to each other according to a predefined criterion. Generally, the Euclidean distance from the the corresponding cluster centers is taken to be the similarity criterion. The problem is to classify the patterns into K clusters such that the sum of the within cluster Euclidean distances over all the clusters will be the minimum. In this article, we describe a clustering algorithm based on a recently developed pipelined genetic algorithm which uses a stochastic selection method.

In pipelined GA the operations of the conventional GAs are performed in several stages in an overlapped fashion. In conventional GA, the operations of selection, crossover and mutation are performed one after another in a generation. Here, selection operation could not be started before all the chromosomes in the parent generation have been evaluated. This is because the conventional selection schemes depend on a complete pool for selecting a single chromosome. For this reason, in pipelined GA, we have used a stochastic selection scheme which is described below.

Here, a chromosome (a cluster configuration)C_i with value $f(C_i)$ is considered from a pool $P(g)$ of generation g, and is selected based on Boltzmann probability distribution function. Let, f_{max} be the fitness value of the currently available best string. If $f(C_i) > f_{max}$, then it is selected and f_{max} is set equal to $f(C_i)$. Otherwise, it is selected with Boltzmann probability $p = exp[-(f_{max} - f(C_i))/T]$ where $T = T_0(1-\alpha)^k$ and $k = \left(1 + 100\frac{g}{G}\right)$; g is the current generation number. G is the maximum value of g. α can be any value in $[0, 1]$, and T_0 may be selected from the interval $[5, 100]$. Thus, T will decrease exponentially with increase in g, and hence the value of the probability p. This ensures convergence. As computation proceeds towards $T = 0$, the final state is reached, i.e., the global solution is achieved.

CLUSTERING USING PIPELINED GA

Unlike conventional selection schemes, the decision on selection of a chromosome is made, once it is evaluated. When a pair of chromosomes has been selected, crossover operation may start without waiting for evaluation of all the others in $P(g)$. After crossover, they can be used for mutation operation, and then be evaluated. Note that, although crossover operation is performed on a pair of chromosomes, mutation and evaluation operations are performed on a single chromosome. When a chromosome is evaluated, it is put into population pool along with its fitness value for selection operation in the next generation $(g + 1)$. Thus the processes corresponding to selection, crossover, mutation and evaluation in a particular generation can work simultaneously, in an overlapped fashion. Since there is high differences between the stage times of the various stages, we require to arrange for multiple units in parallel at different stages. However, we shall use the crossover time to be the unit of time, as this is the smallest of all stage times. Number of selection units will be taken as two, as one crossover requires two selected chromosomes. The multiplicity at the mutation, distribution and evaluation stages depend on the complexity of the data set.

The main objective behind such a configuration of the pipeline is that, we want the average stage times for each chromosome to be one T-cycle. By decomposing the operations of GA into stages of a pipeline we can streamline these operations so that their functioning becomes overlapped in nature.

DESIGN OF PIPELINE

In this section, we try to develop an architectural framework of the GA, by decomposing its operations into stages of a pipeline which can be used to solve the clustering

0-7803-8243-9/04/$17.00 © 2004 IEEE

problem. In crisp clustering we distribute n data points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be represented by S and the K clusters by $\{C_1, C_2, C_3, \ldots, C_K\}$. Then

$$\begin{array}{ll} C_i \neq \emptyset & for \quad i = 1, \ldots, K, \\ C_i \bigcap C_j = \emptyset & for \quad i, j = 1, \ldots, K, \; i \neq j, \text{ and} \\ \bigcup_{i=1}^{K} C_i = S. & \end{array}$$

The problem of clustering is considered to be an optimization problem where we try to minimize a clustering metric, viz., the intra cluster spread. Mathematically, the clustering metric $\mathcal{F}$ given by Eqn. 1.

$$\mathcal{F}(C_1, C_2, ..., C_k) = \sum_{i=1}^{K} \sum_{\mathbf{x}_j \in C_i} ||\mathbf{x}_j - \mathbf{z}_i|| \qquad (1)$$

The goal of clustering is to search for appropriate cluster centers $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K$ such that the clustering metric $\mathcal{F}$ is minimized. GAs can be applied to clustering for getting good solutions in reasonable time. Several such attempts may be found in [2, 3, 4]. However, due to the nature of conventional GAs, the process is likely to be slow, more so when the size of the data is large. In this article our aim is to design a clustering technique which is simple and able to provide good solution fast enough while being independent of the distribution of the data set. This is achieved through the use of pipelined GA.

Pipeline architecture

Since our aim is to solve the clustering problem, we first consider the basic operations in clustering and then combine them with the genetic operators. In case of crisp clustering, we need to identify the cluster centers so that the sum of within cluster Euclidean distances becomes the minimum. The process is thus composed of the following steps.

- Step 1: Initialize cluster centers randomly.
- Step 2: Distribute the elements among the clusters on the basis of minimum Euclidean distance and compute the new cluster centers.
- Step 3: Compute the sum of within cluster Euclidean distances.
- Step 4: Perturb the cluster centers by small amounts.
- Step 5: Repeat Steps 2 to 4 as long as the sum can be reduced.

In order to formulate the above process within the GA framework, we need to maintain a population of probable solutions (strings of cluster centers). The clusters are then formed by distributing each element to the cluster whose center is the closest to it. Each cluster configuration C_i is evaluated for its sum of within cluster Euclidean distances. The fittest candidates are selected for the next generation which then undergo the crossover and mutation operations to generate offspring. The whole process is repeated for a number of generations.

From the above description, we can identify five major functions that are : (i) selection (S), ii) crossover (C), (iii) mutation (M), (iv) distribution (D) and (v) evaluation (E). Thus we can construct a five stage pipeline, each stage corresponding to one of these functions, as shown in Figure 1. It is to be noted here that after distribution of the points the cluster centers are recomputed. This task may be done within the evaluation unit. We need to maintain a buffer memory to reserve the chromosomes after evaluation. This buffer is organized in a FIFO manner. The chromosome that is evaluated first will thus be used at first by the selection stage for the next generation.

As mentioned earlier, the selection operation requires two parallel units so that it can provide two strings to the crossover unit in due time. Mutation and fitness evaluation should be done in multiple units that operate in parallel. The number of units for mutation is determined by the length of a chromosome. It is found that distribution and evaluation are two time consuming processes compared to the other operations, and the number of units for these stages are determined by the complexity of the data set. Here, complexity means the number of elements present in the data set.

Let us assume that a chromosome consists of l number of genes each of which represents a cluster center $\mathbf{z}_i$ $(1 \leq i \leq l)$ as a vector of real numbers. Let, S_t, C_t, M_t, D_t and E_t be the stage times for selection, crossover, mutation, distribution and evaluation operations respectively. Among them, C_t is normally found to be the minimum. We call this minimum time as one T-cycle. Let, $S_t = sC_t$, $M_t = mC_t$, $D_t = dC_t$ and $E_t = eC_t$. Therefore, the ratio of S_t, C_t, M_t, D_t and E_t becomes $s : 1 : m : d : e$. That is, s, m, d and e number of T-cycles are required for selection, mutation, distribution and evaluation operations respectively. Thus for one crossover unit we need, for efficient utilization of resources, $\lceil s \rceil$, $\lceil m \rceil$, $\lceil d \rceil$ and $\lceil e \rceil$ pairs of units for selection, mutation, distribution and evaluation respectively. For sake of simplicity, let us consider, from now on, $s = \lceil s \rceil$, $m = \lceil m \rceil$, $d = \lceil d \rceil$ and $e = \lceil e \rceil$. Here the units are counted in pairs because one crossover needs two selected strings. From the above ratio, it is clear that, if the crossover unit takes 1 unit of time (T-cycle) to perform one crossover, the selection, mutation, distribution and evaluation units take s, m, d and e units of time to perform one selection, mutation, distribution and evaluation operations respectively. Thus for proper and efficient utilization of the resources, we should use s, m, d and e pairs of respective units for one crossover unit.

Speedup

Speedup of a general pipeline is defined as $S = \frac{T_{NP}}{T_P}$ where T_{NP} $(= nk, n =$ number of executions and $k =$ number of stages) and T_P $(= n + k - 1)$ are the computation times (in terms of number of T-cycles) required for non-pipelined and pipelined systems respectively. In the proposed clustering, if appropriate number of units are considered, the average time per chromosome in each stage becomes equal to one T-cycle. This is the ideal hardware configuration. However, we can use less number of units at the stages. Let, for any arbitrary configuration, m_n, d_n and e_n be the number of pairs of units used at mutation, distribution and evaluation

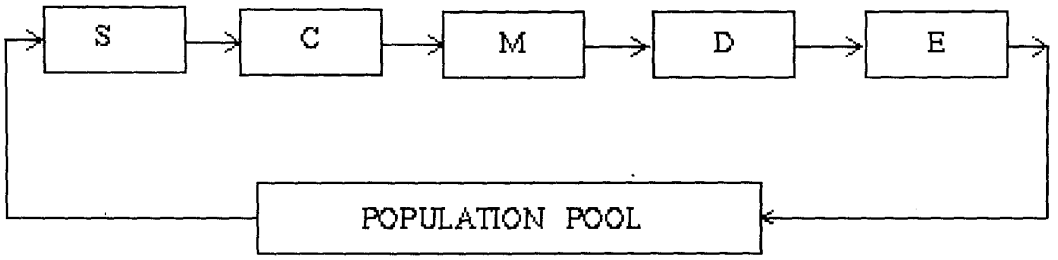

Figure 1. Pipeline stages for the GA-clustering

stages corresponding to one crossover unit and one pair of selection units. In our case, n = population size $\times$ number of generations.

Consider a pipeline where $s = 1$, $m = 4$, $d = 8$ and $e = 6$. Here we require $s+1+m+d+e = 20\,T$-cycles to get the first pair of children chromosomes. Since n = population size $\times$ number of generations, we may assume, without loss of generality, n to be equal to the population size executed for one generation only. After obtaining the first pair of children, the remaining children will come out in pairs at each successive T-cycles. Therefore, the number of T-cycles required for the remaining pairs is $\left(\frac{n}{2}-1\right)$. Thus, the total number of T-cycles required for the pipeline is $T_P = 19 + \frac{n}{2}$ For a non-pipelined system configured with the same multiplicity of stages (as that of the pipelined one), the number of T-cycles considering all the five stages sequentially is

$$T_{NP} = \frac{n}{2} + \frac{n}{2} + \frac{n}{2} + \frac{n}{2} + \frac{n}{2} = \frac{5n}{2}.$$

So, the speedup attained is

$$S = \frac{T_{NP}}{T_P} = \frac{\frac{5n}{2}}{\frac{n}{2}+19}$$

for situations when $n >> 19$, $S \approx 5$. This is the ideal speedp.

As mentioned earlier, we can use less number of units at mutation and evaluation stages. Let, for any arbitrary configuration, m' and e' be the number of pairs of units used at mutation and evaluation stages corresponding to one crossover unit and one pair of selection units. Here, $m' < m$ and $e' < e$, i.e., the number of units at the mutation and evaluation stages are less than that needed for full multiplicity of these stages. Let $r_m = \lceil \frac{m}{m'} \rceil$ and $r_e = \lceil \frac{e}{e'} \rceil$, i.e., r_m and r_e are the factors by which multiplicity is reduced at the corresponding stages. We define the *reduction factor* for a pipeline as the maximum of r_m, r_d and r_e, i.e., *reduction factor*, $r = max(r_m, r_d, r_e)$. When $r = 1$, the pipeline has full multiplicity and is referred to as a *full pipeline*. For $r > 1$, it is called a *reduced pipeline*.

Now, let us a consider a reduced pipeline where r_m, r_d, r_e and r represent reduction factors for mutation, distribution, evaluation stages and that for the whole pipeline. By definition, at least one of r_m, r_d and r_e is equal to r and the others are less than or equal to r. For such a reduced system we get

$$\begin{aligned} T_P &= 1+1+m+d+e+\left(\frac{n}{2}-1\right)\times r \\ &= 2+m+d+e+\left(\frac{n}{2}-1\right)\times r \end{aligned}$$

and

$$T_{NP} = \frac{n}{2} + \frac{n}{2} + \frac{n}{2} \times \lceil \frac{m}{m'} \rceil + \frac{n}{2} \times \lceil \frac{d}{d'} \rceil + \frac{n}{2} \times \lceil \frac{e}{e'} \rceil.$$

If the pipeline is a uniformly reduced one with $r_m = r_d = r_e = r$, we have

$$T_{NP} = \frac{n}{2} + \frac{n}{2} + \frac{n}{2} \times r + \frac{n}{2} \times r + \frac{n}{2} \times r.$$

Now, in our example system, if the reduction factor be $r = 2$, then we get

$$T_P = 16 + \left(\frac{n}{2}-1\right) \times 2 = 14 + n$$

and

$$T_{NP} = \frac{n}{2} + \frac{n}{2} + \frac{n}{2} \times 2 + \frac{n}{2} \times 2 + \frac{n}{2} \times 2 = 4n.$$

Therefore, speedup $S \approx 4$ for $n >> 14$. From the above discussion it is clear that a fixed hardware setup is also possible. Note that in the non-pipelined system, termination of the process is checked at the end of each generation. On the other hand, for pipelined system, once the pipeline is activated, there is no necessity of such termination checking.

EXPERIMENTAL RESULTS

Two artificial data sets Data1 and Data2, and two real life data sets Iris and Crude Oil are used here. Data1 is a three dimensional data set of 40 elements distributed in four classes. Data2 is a two dimensional data set of 250 elements distributed in five classes. The real life data sets used are Iris and Crude Oil data with 150 ad 56 data items in 4 and 5 dimensions respectively. Both have 3 clusters. We have considered population size, $L = 50$, $P_c = 0.6$, $P_m = 0.05$ and $\alpha = 0.05$. T_0, is set to 50. The stage times given in Table 1 are proportional to the time required for one scheduled operation. Stage times of mutation, distribution and evaluation functions increases with increase in the number of dimensions and number of clusters (Table 1). Thus the number of mutation,

Table 1. Stage times of PLGA for crisp clustering and the corresponding speedups

Data Set	Stage Times					Speedup obtained with	
	S_t	C_t	M_t	D_t	E_t	*Full Muliplicity*	*Fixed H/W Setup*
*Data*1	1	1	1	56	13	4.93	3.94
*Data*2	1	1	1	298	52	4.67	2.28
Iris	1	1	2	208	60	4.74	2.29
CrudeOil	1	1	1	97	29	4.88	3.90

distribution and evaluation units to be used depend on the data. For $Data1$, as seen from Table 1, the number of T-cycles needed for different stages may be considered to be in the ratio of 1:1:1:56:13. Therefore, for clustering this data set, we can use 2 S-units, 1 C-unit, 2 M-units, 112 D-units and 26 E-units.

Speedups are calculated considering $n = 10,000$ and a non-pipelined system where stages have the same multiplicity as those for the pipelined one. Considering sufficient amount of hardware components are available for each of the stages we can satisfy the condition that one pair of strings will come out from each stage, on the average, in one T-cycle. Then we can attain maximum speedup. Otherwise, if we have a fixed hardware setup, for example, with 1 pair of S unit, 1 C unit, 10 pairs of M units, 100 pairs of D units and 50 pairs of E units, speedup will have a lower value. Speedups obtained in both the situations are shown in Table 1.

Higher computation time requirements at the D and E stages may cause a problem to attain streamlined operation of the pipeline. Because, we can resolve the computation time mismatch of different stages by allocating parallel units, but we also need to use a huge initial population. The actual size of the initial population needed for continuous operation of the pipeline (full pipeline) is $S_t + C_t + M_t + D_t + E_t$ pairs. However, this problem can be tackled by designing proper hardwares for the D and E stages which can exploit high parallelism inherent within the corresponding operations.

CONCLUSION

A pipelined genetic algorithm based method for clustering has been described in this article. It has been shown that when proper multiplicity of different stage units are used, a maximum speedup of 5 is attainable compared to conventional GAs executed serially using similar multiplicity of stage units. However, if we consider speedup compared to a uniprocessor executing GAs serially, it will be much more. By use of proper hardware, one can develop an extremely fast version of the GA-clustering. The authors are working to develop one such hardware pipeline.

REFERENCES

[1] Tou, J. T. and Gonzalez, R. C. Pattern Recognition Principles. Addison-Wesley, Reading, 1974.

[2] Maulik, U. and Bandyopadhyay, S. "Genetic algorithms based clustering technique", Pattern Recognition, vol. 33, pp. 1455–1465, 2000.

[3] Murthy, C. A. and Chowdhury, N. "In search of optimal clusters using genetic algorithms", Pattern Recognition Letters, vol. 17, pp. 825–832, 1996.

[4] Pakhira, M. K., Bandyopadhyay, S. and Maulik, U. "Fuzzy genetic clustering and application to pixel classification of satellite images", Proc. of IEEE Technical Conference, IEEE TENCON-2003, Bangalore, India, 2003, pp. 872-876.