A New Parallel Genetic Algorithm

Ling Tan David Taniar Kate A. Smith
Monash University
School of Business Systems
Clayton, Vic 3800
AUSTRALIA
Email: {Ling.Tan, David.Taniar, Kate.Smith}@infotech.monash.edu.au

Abstract

One problem of propagating the globally fittest individual via neighbourhood evolving in both island model and cellular model of existing parallel genetic algorithms (PGA) is that the migration of globally best individual is delayed to non-adjacent processors. That may cause inferior search in those sub-populations. The propagation delay of the globally best individual is proportional to the network distance between two processors. Delayed migration of best individual in parallel genetic algorithms is an essential deviation from sequential version of genetic algorithm, in which the best individuals are always used to compete with other individuals. To solve this problem, this paper proposes an extended version of island parallel genetic algorithm, Virtual Community PGA (VC-PGA). In this paper, VC-PGA is applied in a case study of optimizing parameters of back-propagation neural network classifier.

1 Introduction

Two major parallel genetic algorithms (PGA) are reported in literature, namely island model and cellular model. *Island model* (or network model) runs an independent GA with a sub-population on each processor, and the best individuals in a sub-population are communicated either to all other sub-populations or to neighbouring population [2, 3]. *Cellular model* (or neighbourhood model) runs an individual on each processor, and cross with the best individual among its neighbours [4]. The problem of migrating globally best individual via neighbourhood evolving is that the best individual is not immediately used in the subpopulations of non-adjacent processors. This delay causes inferior search in those sub-populations. Delayed migration of best individual in parallel genetic algorithms is an essential deviation from sequential version of genetic algorithm, in which the best individuals are always used to compete with other individuals throughout the life circle of iterations. To solve this problem, we propose an extended version of island parallel genetic algorithm, *Virtual Community PGA* (VC-PGA).

This rest of paper is organized as follows. Section 2 gives brief introduction on genetic algorithm (GA), parallel genetic algorithm (PGA), and describes GA-guided parametric optimisation on back-propagation neural network classifier, which will be used as a case study of VC-PGA. Section 3 describes the proposed virtual community model (VC-PGA) in details. Section 4 describes the implementation issues VC-PGA. Section 5 describes parameters and environment of performance studies and discusses some initial results obtained, which is then followed by the conclusions in Section 6.

2 Background

In this section, we briefly describe genetic algorithm (GA), existing parallel genetic algorithms (PGA) and their problems. Then, we introduce parametric optimisation of neural network classifier, which will be used as a case study in our experiments later.

2.1 Genetic Algorithm (GA)

GA is a domain-independent global search technique. It works with a set of solutions that are encoded in strings and a fitness function to evaluate these solutions. These strings represent points in search space. In each iteration (or generation), a new set of solutions is created by crossing some of good solutions in current iteration and by occasionally adding new diversity of search [6]. This

Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'02)
1087-4089/02 $17.00 © 2002 **IEEE**

process is done through three important operators, i.e. *selection*, *crossover*, and *mutation*. In the following section, we describe each operator in turn.

Selection. Candidate selection is based on evaluating the fitness of candidates. Candidate fitness is calculated though objective function. Fitness values generally need to be scaled for further use. Scaling is important to avoid early convergence caused by dominant effect of a few strong candidates in the beginning, and to differentiate relative fitness of candidates when they have very close fitness values near the end of run [1]. Three basic methods are proportional selection, tournament selection, and ranking selection. In proportional approach, selection is based on a probability of candidate fitness in a population, e.g. roulette wheel method. In tournament approach, selection is based on sub-group competition of randomly selected with or without replacement. In ranking approach, selection is based on fitness of individuals, and the rank N is assigned to the best individual and the rank 1 is to the worst [8].

Crossover. Crossover is the most important operator of GA. It explores new search space by recombining existing search elements, which may produce better solutions based on partial and local solutions. In a simple crossover, the operation takes two candidates and recombines them at a random point. Crossover rate is generally set with a high probability.

Mutation. Mutation is an operator to explore new search space by introducing new search elements not found in existing candidates. In a binary string, the operation flips a single bit. Mutation rate is generally set with a very low probability.

By iteratively applying these operators, GA-based search is able to converge on one of global optimal. GA is a powerful technique to solve optimization problems in a large search space, where random walk is not feasible.

2.2 Parallel Genetic Algorithm (PGA)

Two major parallel versions of GA (PGA) are reported in literature, namely island model and cellular model. *Island model* (or network model) runs an independent GA with a sub-population on each processor, and the best individuals in a sub-population are communicated either to all other sub-populations or to neighbouring population [2, 3].

Cellular model (or neighbourhood model) runs an individual on each processor, and cross with the best individual among its neighbours [4]. Cellular model can be seen as a massive parallel extension of island model where population is reduced to a single individual on each processor.

Figure 1. The Degree of Inferiority

The problem of neighbourhood evolving, in both *island model* and *cellular model*, is that the migration of globally best individual is delayed to non-adjacent processors. That may cause inferior search in those sub-populations, which are not physically adjacent to the sub-population where the best individual is found. The propagation delay of the globally best individual is proportional to the distance between the processor which sends the globally best individual and the processor which receives it. The degree of inferiority intensifies circularly outwards as shown in Figure 1. Delayed migration of best individual in parallel genetic algorithms is an essential deviation from sequential version of genetic algorithm, in which the best individuals are always used to compete with other individuals.

In addition, it is not scalable to use multicast to communicate local best individuals among sub-populations in a network environment (e.g. cluster of workstations). To solve the above problems, we propose an extended version of parallel genetic algorithm, *Virtual Community PGA* (VC-PGA) in section 3.

2.3 GA-guided Parameter Optimisation in Neural Network

This section describes details of GA-based search in application to parameter optimization in a back-propagation neural network classifier (NN).

Data mining is a process of extracting useful information from data. To use any data mining algorithm, a set of parameters often needs to be specified. Generally, a set of default values is provided for a given data mining algorithm, and these default parameter values were considered a 'good' estimate for any data set. However, data mining models are known to be data dependent, and so are for their parameter values. Default values may be

good estimates for some data sets, but they are often not tuned to a particular data set.

Parametric search scans through parameter space in order to find the best set of parameter values. However, parameter space is often very large, and brute-force search is simply too expensive to be feasible. This paper investigates parallel genetic algorithm based heuristic search techniques to optimise parameters of data mining algorithms. As a case study, back-propagation neural network (NN) will be used for parameter optimisation in this paper.

The reason we choose to use NN in our parameter optimization case study is that (*i*) parameter value of NN is difficult to choose, and default values are not always good estimates; and (*ii*) it has a relatively large parameter space, which is too expensive to enumerate. A number of parameters need to be specified for a back-propagation NN. The choice of single parameter value and the combining effects of different parameter values have an influence over prediction accuracy of the final data mining model. Parameters in a back-propagation NN include learning rate, momentum, the number of hidden layer and hidden node, and epoch; and their overall search space is 2^{34}.

3. The Proposed PGA: Virtual Community Model (VC-PGA)

In this section, we describe in details the proposed virtual community parallel genetic algorithm (VC-PGA). VC-PGA aims to solve two problems in the existing parallel genetic algorithms explained in section 2.2: *i*) delayed migration of globally fittest individual. *ii*) communication overhead introduced by multicast. VC-PGA addresses these two problems with the concept of virtual community. The globally best individuals are migrated to all processors in the networked environment through hierarchy of virtual communities.

In virtual community model, each processor hosts a sub-population just like in island model. A *local community* (LC) is formed among neighboring processors, and each local community has a server processor to facilitate exchanges of best individuals within community and with other communities as shown in Figure 2. To facilitate exchange of best local individuals across community, *virtual community* (VC) can be formed and a virtual server processor performs similar tasks as a local community server processor.

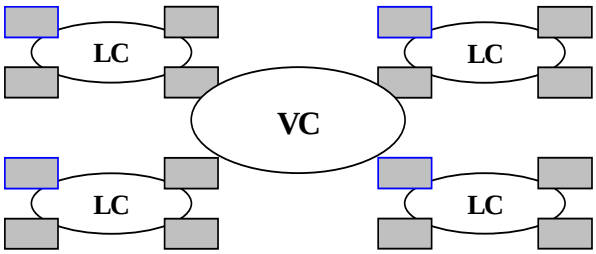

Figure 2. The Proposed VC-PGA Model

Virtual community model gives three advantages over island model: (*i*) Local sub-populations can get globally fittest individuals, (*ii*) The communication overhead is much less expensive, and (*iii*) The evolution surface of solution space is independent from topology of network of workstations. The procedure of VC-PGA is described as follows:

Step 1: Set up local communities and virtual communities in a cluster of workstations. A local community is formed with adjacent processors, and one of processor is elected as its community-server. Grouping local community-servers forms a virtual community. Higher-level of virtual communities can be formed if necessary.

Step 2: Initialise a sub-population of size *p* on each community member across network.

Step 3: On each processor of a local community, run GA for a fixed number of iterations, and send top *n* fittest of local sub-population to the community-server. At the mean time, compare the fitness of any received individual with the best local fitness, and include it to local population if the received has a higher fitness value.

Step 4: On each community-server, find the best individuals of local community from the received best individuals of community members. Send top *n* fittest individuals of local community to higher level community server. At the mean time, compare the fitness of any received individual from its virtual community with the best local fitness, and include it to local population if the received has a higher fitness value. Send the best individuals to community members.

4. Implementation issues

In this section, we describe implementation issues of VC-PGA. The implementation of VC-PGA is based on its sequential version. Therefore we first describe GA issues and then followed by issues specific to virtual community model of PGA.

4.1 GA Implementation

Like any other GA applications, two essential problems have to be addressed before using GA to parameter optimisation.

First is string encoding. Many encoding schemes have been proposed, for example, integer coding and grey coding. There is no standard way to choose these schemes and the choice really depends on the nature expression of problems. The order of putting encoded parameter into a string is also important. In principle, closely associated parameters need to be put together to avoid disruption caused by crossover operator. For parameter optimisation problem, we use binary encoding. In addition, we assume no knowledge on relationship of parameters, and parameters are encoded in a random order.

The second issue is how to evaluate string. The fitness of string is evaluated though prediction accuracy of NN model based on a data set. The fitness function is the reverse of classification error (i.e. root mean square error). In the following section, we illustrate how GA and its parallel version can be used in our performance study.

There are many variations of GA operators. Our purpose is to see the effectiveness of GA-guided parameter optimization, rather than to compare different GA operators. Therefore, we use the simple genetic algorithm (SGA) described in [1] for our study of both GA and PGA. However, we use elitism, which always include the best search point from the last iteration to the current iteration, to prevent early convergence. The procedure of SGA is described as the following.

Step 1: Initialise population.

Step 2: For a specified number of iteration, do selection, crossover and mutation according to a set of user specified parameters. And then evaluate population individuals for next iteration.

GA is implemented in five classes in Java language, i.e. *parameter*, *chromosome*, *generation*, *evaluationFunction*, and *ga*.

Parameter class is a container of single parameter. It allows users to specify a parameter value and its range in the form of [min, max], and it scales the value down into unsigned integer.

Chromosome class consists of attributes and methods of binary string. It encodes multiple parameter values into binary values, and joins them into a single string. It also has a decoding operation to reverse the process.

Generation class applies three GA operators on a population of chromosomes (or individuals). As part of pre-processing of current generation, scaling of fitness is applied to entire population so that two objectives can be achieved, i) to avoid early convergence caused by a few relatively fitter individuals at beginning of run, and ii) to avoid convergence caused by dominance of a large portion of highly-fit individuals at end of run. In selection, a deterministic sampling method is implemented. To select a population, first compute expected value of individuals for each chromosome. Then allocate individuals based on integer part of expected number. If population is not filled up after initial allocation, sort previous population on fractional part of expected value and fill up the gap from the top of sorted list. Include the best individual at this point in the new population. Once selection is done, select a list of chromosomes for crossover based on specified crossover probability. For each pair of chromosomes, cross over at a randomly chosen point. Then perform mutation by negating each bit of string based on pre-specified probability.

EvaluationFuction class evaluates a set of parameter values on back propagation NN classifier and returns its fitness value. In order to evaluate these parameter values, a training data set is required. Then build classifier model, evaluate it with n-fold cross validation, and return fitness value as reverse value of evaluation error rate.

Ga class allows users to specify various parameters of GA, e.g. cross over rate, and mutation rate, and iteration number. And it initialises a generation instance, and iterates the generation till its stop criteria.

After GA is implemented, its performance is compared with random search in same environment setting. A separate class, *RandomSearch*, is implemented for this purpose. The class iteratively calls an instance of *EvaluationFunction* class with randomised parameter values. To make search results comparable, multithreading is used in our implementation so that each search method is running on a separate thread. To record results at end of fixed number of iteration, synchronization between two threads makes use of global static variables as signals. A duplicate copy of data set is used to avoid I/O race.

4.2 VC-PGA Implementation

In the above section we described how GA is implemented in Java language and how GA-based search is compared with random search. In this section, we describe VC-PGA implementation details.

One issue in VC-PGA is how to form local communities (LCs) and virtual communities (VCs). By definition, a local community refers to a group of adjacent workstations which includes a leader station and member stations, and a virtual community is a special case of local community which consists of a group of leader stations. A workstation is included in a community if it is the closest to that community than other communities. This is very similar to clustering problems in data mining. A data mining clustering algorithm identifies sparse and dense regions in a large set of multi-dimensional data points. In this case, local community and virtual community can be taken as clusters, and community leaders are the centroids (the centres of clusters). The distances between any two workstations are measured in term of round-trip time. First local communities of given workstations need to be found by applying a clustering algorithm, and then virtual communities of leader stations can be found by running the algorithm again, and higher level of virtual communities can be located in same way. We implemented k-means clustering algorithm [7] for its simplicity, and the procedure is described in the following section.

Step 1: Find distances between any two processors.

Step 2: Specify a cluster number, k, and randomly select k processors as initial cluster centres.

Step 3: Processors are assigned to their closest cluster centres (processors) based on Euclidean distance function.

Step 4: Compute the centroid of each cluster. These centroids are the new cluster centres.

Step 5: Repeat Step 3) and 4) until these centroids remain the same in consecutive runs. Thus, a cluster of workstations with k local communities is set up. Centroids are community servers, and non-centroids processors in the cluster are community members.

Once we partitioned workstations with a clustering algorithm, simply install GA on each workstation with a sub-population. The communication between a leader station and its member stations are done via client-server programming.

5 Performance Evaluation

5.1 Environment and Parameters

Performance study of PGA-based parameter optimization is conducted over local area network, which consists of eight homogeneous Pentium workstations. Back-propagation neural network classifier in WEKA [5] data mining library is used to build and evaluate classification models. Fitness value is derived from root mean square error returned by neural network classifier. Parameters and their value ranges used in PGA-guided parametric optimization of back propagation neural network classifier are summarized in Figure 3. Default values in Figure 3 refer to default values used in standard back-propagation Neural Network classifier, and included here for cross reference. After iterations of PGA-guided search, parameter values derived from the search result are the tailored values for the particular data set.

Parameters	Value Range	Bit Length	Default Value
Learning Rate	[0, 1.00]	7	0.3
Momentum	[0, 1.00]	7	0.2
Hidden layer	[1, attribute number]	Round($\log_2$ A)	4
Epochs in training	[1, 1000]	10	500
Error threshold in validation	[1, 100]	7	30

Figure 3. Parameters in Back propagation NN Classifier

Proportional selection with elitism is used in PGA, and a clustering algorithm is used as pre-processing in PGA to partition workstations by distance. For PGA-related parameters, a high cross over rate and a low mutation rate are used. To study speed-up of PGA, we keep total population size constant, and decrease population size on each workstation while increasing the number of workstations. Parameters in VC-PGA are summarized in Figure 4.

Parameters	VC-PGA
Generation	20
Population size	15, 20, 30, 60, 120
Cross over rate	0.8
Mutation rate	0.01

Figure 4. Parameters in GA and VC-PGA

5.2 Performance Analysis

We implemented VC-PGA model on up to 8 stations in a local area network to observe its speed-up. The depth of VC hierarchy is one, and two LCs are formed. To obtain speed-up, we keep total population size and the number of iterations constant, while using a different cluster size from 1 to 8. For example, population size is 60 on each station in a 2-station cluster, population size 30 on each station in a 4-station cluster and so on. The elapsed time for each cluster to complete its job is taken using the maximum time spent across all stations in a cluster at end of iteration. As shown in Figure 5, VC-PGA closely follows a linear speed-up.

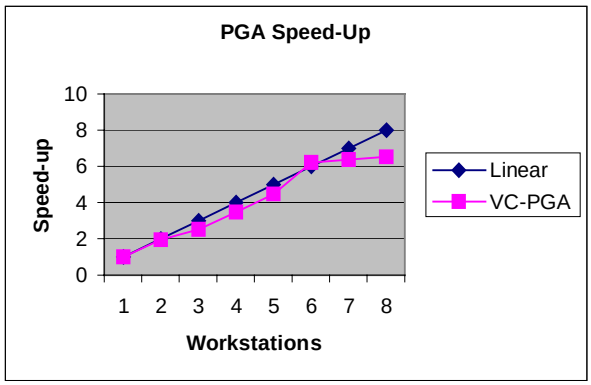

Figure 5. PGA Speed-up

Figure 6 shows population fitness in speed-up of VC-PGA parametric optimization. Best fitness found at all cases is similar to each other. It indicates that search outcomes are comparable among different experiments when more workstations are used to process the same job size. Average population fitness across all sub-populations seems to increase as more nodes are used to work with smaller population size. This is because the results are taken at end of relatively small number of generation where sub-populations have not converged.

Nodes	Population Size	Average Fitness	Best Fitness
1	120	70	77.66264
2	60	71	77.51973
4	30	73.5	77.52943
6	20	73.3	78.20998
8	15	75.3	77.51269

Figure 6. Population Fitness in PGA Speedup

6 Conclusions

This paper proposed an extended version of island model, namely *Virtual Community PGA* (VC-PGA), to solve the problem of delayed propagation of the globally fittest individual via neighbourhood evolving in both *island model* and *cellular model*. Delayed migration of best individual in parallel genetic algorithms is an essential deviation from sequential version of genetic algorithm, in which the best individuals are always used to compete with other individuals. In our future work, we would like to investigate VC-PGA in more details including scalability and performance comparison with existing PGA models.

Acknowledgment

The authors would like to thank anonymous referees for their helpful comments.

References

[1] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Co., 1989.

[2] T. Starkweather, D. Whitley, and K. Mathias, "Optimization using distributed genetic algorithm", *Parallel Problem Solving from Nature*, pp. 176-185, Springer Verlag, 1991.

[3] M. Gorges-Schleuter, "Explicit parallelism of genetic algorithms through population structures", *Parallel Problem Solving from Nature*, pp 150-159, Springer Verlag, 1991.

[4] V. S. Gordon and D. Whitley, "A Machine-Independent Analysis of Parallel Genetic Algorithms", *Complex Systems*, 8:181-214, 1994.

[5] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, 2000.

[6] S. M. Sait and Y. Youusef, "Iterative Computer Algorithms with Application in Engineering", *Solving Combinatorial Optimization Problems*, IEEE Computer Society, 1999.

[7] L. Kaufman and P.J. Rousseeuw, *Finding Groups in Data: an Introduction to Cluster Analysis*, John Wiley & Sons, 1990.

[8] T. Blickle and L. Thiele, "A Comparison of Selection Schemes used in Evolutionary Algorithms", *Evolutionary Computation*, volume 4, number 4, pp.361-394, MIT-Press, 1996.