Middle-East Journal of Scientific Research 15 (12): 1651-1654, 2013
ISSN 1990-9233
© IDOSI Publications, 2013
DOI: 10.5829/idosi.mejsr.2013.15.12.2

Coarse-Grained Parallel Genetical Gorithm to Solve the Shortest Path Routing Problem Using Genetic Operators

[1]*R. Udayakumar,* [2]*K.P. Thooyamani and* [1]*V. Khanaa*

[1]School of Computing Science, Bharath University, Chennai-600073, India
[2]Information Technology Bharath University Chennai-600073, India

Abstract: In computer networks the routing is based on shortest path routing algorithms. Based on its advantages, an alternative method is used known as Genetic Algorithm based routing algorithm, which is highly scalable and insensitive to variations in network topology. Here we propose a coarse-grained parallel genetic algorithm to solve the shortest path routing problem with the primary goal of computation time reduction along with the use of migration scheme. This algorithm is developed and implemented on an MPI cluster. The effects of migration and its performance is studied in this paper.

Key words: Coarse Grained Components % Mpi Cluster % Genetic Operators % Parallel Genetic Algorithm % Shortest Path Routing.

INTRODUCTION

In computer networks, Cthe task of finding a path from source node to destination node is known as routing. For a given network, it consists of more than one path. Based on the shortest path, we have to find routing to a given network. Examples of such algorithms are Dijkstra‘s & BellmanFord algorithms. The alternative methods for shortest path routing algorithms have been find out by researchers. One such alternative method is the use of Genetic Algorithm, which is a multi-purpose search & optimization algorithm. Here it encodes the problem into a chromosome which has several genes and a group of chromosomes referred as a population is represented as a solution to this problem. For every iteration, the chromo somes in population will undergo one or more genetic operations such as crossover and mutation.The result of the genetic operations are the next generations of the solution. The process continues until a solution is found or a termination condition exists. The idea behind genetic algorithm is to have the chromosomes in the population to slowly converge to an optimal solution [1].

At the same time, the algorithm is supposed to maintain enough diversity so that it can search a large search space. Based on these two characterstics, Genetic Algorithm is called as a good search and optimization algorithm.

The earliest Genetic Algorithm-based algorithm was proposed by Munetomo, to generate alternative paths in case of link failures. In the proposed algorithm, the algorithm chromosome is encoded as a list of node id‘s from source to destination path. The chromosomes will be of variable length because different paths can have different number of nodes. This algorithm employs crossover, mutation and migration genetic operators in generating the next generation of solutions. Chang also proposed a GA-based routing algorithm which is similar to Munetomo’s algorithm but differs in its implementation. The algorithm proposed by Chang had several advantages, the first is that GA is insensitive to variations in network topologies with respect to route optimality and convergence speed. The second is that GA-based routing algorithm is scalable which means that the real computation size does not increase very much as the network size gets larger. However genetic algorithm is not fast enough for real time computation. In order to achieve a really fast computation time in genetic algorithm, it requires a hardware implementation. In this paper, we propose a parallel genetic algorithm for the

Corrseponding Author: R. Udayakumar, School of Computing Science, Bharath University, Chennai-600073, India.
E-mail: rsukumar2007@hotmail.com.

shortest path routing problem. The notion behind this is, parallel implementation of genetic algorithm should improve its computation time and it can be implemented on a MPI (message passing interface) cluster [2].

Existing Genetic Algorithms: Generally genetic algorithm will find good solutions in reasonable amount of time, but increases in time to find solutions if they are applied to harder and bigger problems. To overcome this problem we will go for parallel implementation of genetic algorithm. The PGA consists of multiple computing nodes, those depends on type of PGA used. [3] There are 4 major types of PGA's, they are master-slave, coarse- grained, fine-grained & hierarchical hybrids.

Master-Slave GA: In Master-Slave GA, one computing node will be the master and the others are slaves. The master node is responsible to hold the population and performs most of the genetic algorithm operations. The master will assign one or more computing intensive tasks to slaves by sending one or more chromosomes to them and it would then wait for the slaves to return their results.

Coarse-Grained GA: In Coarse-Grained, the population is divided into computing nodes which have a sub-population and executes genetic algorithm on its own. The nodes will exchange chromosomes with each other ensuring that good solutions can be spread to other nodes. This exchange can be called as migration where a node sends its best chromo some to other nodes. The other nodes which are having the worst chromosomes will be replaced by the received ones.

Fine-Grained GA: In Fine-Grained, each computing node only has a single chromosome and are arranged in a spatial structure. Here each node communicates only with other neighbouring nodes and the population is the collection of all the chromosomes in each node. To execute a genetic operation, a computing node must interact with its neighbours [4]. The good traits of a superior individual can be spread to the entire population due to the overlapping of neighbourhoods.

Hierarchical Hybrids: This is the final PGA type which is structured in two levels. It operates as a coarse-grained in higher level and as a fine-grained in lower level. Among the four types of PGA's, the fine-grained genetic algorithm has the highest level of parallelism and also has a large communication overhead because of high frequency of interactions between neighbouring nodes.

Proposed Parallel Genetic Algorithm: The coarse-grained genetic algorithm has been choosen for proposed algorithm due to the use of MPI cluster. Since the MPI environment will have a large overhead of communication between the computers. So we go for coarse-grained genetic algorithm which has lowest communication overhead when compared to other PGA types. In this PGA implementation, all the computing nodes randomly creates their own subpopulation and each of them will execute genetic algorithm on its own. One of the computing nodes will be assigned special task to gather results from all the other nodes and then choose the best result as the output of parallel genetic algorithm. This node is called as the collector node.In proposed algorithm, each chromosome is encoded as a series of node id's that are in the path from source to destination. The first gene in the chromosome is always the source and the last gene in the chromosome is always the destination. Since different paths may have different number of intermediate nodes, the chromosomes will be of variable length. However, the maximum length of a chromosome cannot exceed the total number of nodes in the network. Any repeated nodes in the chromosome signify that the path represented by the chromosome contains a loop and in network routing, any loop should be eliminated [5].

The genetic operations used here can be described as follows:

Selection: Selection is used to choose the parent chromosomes for the crossover operation. The selection scheme used in the algorithm is the pair wise tournament selection. In this selection scheme, a parent for the crossover operation is selected by randomly choosing two chromosomes from the population. The one with the higher chromosome between the two will be selected as a parent. To select two parents, this operation is performed twice.

Crossover: Crossover is performed on the two parent chromosomes selected using the selection scheme described above. To ensure that the paths generated by the crossover operation are still valid paths, the two chromosomes selected must have atleast one common node other than the source and destination nodes. If more than one common node exists, one of them will be randomly choosen with equal probability. The choosen node is called the crossover point. For example, assume that we have the following parent chromosomes: Parent chromosome 1=[A B C G H I X Y Z]

Parent chromosome 2=[A K L M I T U Z] Where A and Z are the source and destination nodes respectively. In this example the common node is node I. Therefore, crossover operation will exchange the first portion of chromosome 1 with the second portion of chromosome 2 and viceversa. As a result, the following child chromosomes will be generated: Child chromosome 1:[A B C G H I T U Z] Child chromosome 2:[A K L M I X Y Z] These two chromosomes would then become new members of the population.

Mutation: Each chromosome produced by the crossover operation has a small chance to be mutated based on the mutation probability. For each chromosome that is choosen randomly, with equal probability, among the intermediate nodes in the path from sender to receiver. Once the mutation point is choosen, the chromosome will be changed starting from the node after the mutation point and onwards. For example assume that the following chromosome has been choosen to be mutated.

Original chromosome:
[A C E F G H I Y Z]

where A and Z are the sending and receiving nodes respectively. Assume also that the node G has been choosen as the mutation point. The mutated chromosome would become like this: Mutated chromosome:

[A C E F G x_1,x_2,x_3... Z]

The mutated chromosome now contains a new path from G to Z where x_i is the ith new node in the path. The new path is generated randomly; the same way as the paths in the initial population is generated. The algorithm used to generate the random paths is as follows:

- Start fro the source node.
- Randomly choose, with equal probability, one of the nodes that are connected to the current node.
- If the choosen node has not been visited before, mark that node as the next node in the path. Otherwise, find another node.
- If all the neighbouring nodes have been visited, go back to step1.
- Otherwise, repeat step2 by using the next node as the current node.
- Do this until the destination node is found.

The computing nodes perform the following operations:

- Randomly initialize the starting sub-population.
- Perform the fitness of each chromosome in the population.
- Create mating pool which has all the chromosomes in current population.
- Now apply crossover & mutation operators.
- Goto step2 until the sub-population converges.
- Now send the best chromosome to collector node.

The collector node performs the following steps:

- Receive the best chromosomes from each of the computing nodes.
- Select the best chromosome from the received one‘s, then that would become the shortest path found by parallel algorithm.

The advantage of the proposed algorithm can be more clearly seen by comparing the computation time required by both the proposed PGA and the non-parallel version of the algorithm to get the same accuracy. This is done by running the non-parallel version of the algorithm several times, each for different population size

CONCLUSION

This paper proposed a coarse grained parallel genetic algorithm for solving the shortest path routing with the primary goal of achieving faster computation speed. The experiment is based on MPI cluster environment [6-10]. The accuracy and computation time of the algorithm is dependent on population size and the number of computing nodes. The accuracy decreases linearly with larger number of computing nodes whereas the computation time decreases exponentially. The accuracy and execution time are higher when population size is more. Due to increase in population the computation time increases and this can be reduced by using larger number of computing nodes. When compared to non-parallel version of the same algorithm, the developed parallel algorithm is observed to have a greatly reduced computation time.

REFERENCES

1. Munetomo, M., N. Yamaguchi, K. Akama and Y. Sato, 2001. Empirical investigations on the genetic adaptive routing algorithm in the Internet", in Proc.Congress on Evolutionary Computation, 2: 1236-1243.
2. Chang Wook Ahn and R.S. Ramakrishna, 1999. A genetic algorithm for shortest path routing problem and the sizing of populations?, IEEE Transactions on Evolutionary Computation, 6(6): 1-7.
3. Meghanathan, N. and G.W. Skelton, 2007. Intelligent transport route planning using parallel genetic algorithm and MPI in high performance computing cluster?, in Proc. International Conference on Advanced Computing and Communications, pp: 578-583.
4. Erick Cantu, Paz and D.E. Goldberg, 2001. Efficient and Accurate Parallel Genetic Algorithms, Kluwer Academic Publishers.
5. Kurose, J.F. and K.W Ross, 2007. Computer Networking: A Top-down Approach, Addison-Wesley, 4th Edition.
6. Nahed, M.A., Hassanein, Roba M. Talaat and Mohamed R. Hamed, 2008. Roles of Interleukin-1 (Il-1) and Nitric Oxide (No) in the Anti-Inflammatory Dynamics of Acetylsalicylic Acid Against Carrageenan Induced Paw Oedema in Mice, Global Journal of Pharmacology, 2(1): 11-19.
7. Panda, B.B., Kalpesh Gaur, M.L. Kori, L.K. Tyagi, R.K. Nema, C.S. Sharma and A.K. Jain, 2009. Anti Inflammatory and Analgesic Activity of Jatropha gossypifolia in Experimental Animal Models, Global Journal of Pharmacology, 3(1): 01-05.
8. Parmar Namita, Rawat Mukesh and J. Kumar, 2012. Vijay Camellia Sinensis Green Tea. A Review Global Journal of Pharmacology, 6(2): 52-59.
9. Jagadeeswaran, M., Gopal, B. Jayakar and T. Sivakumar, 2012. Simultaneous Determination of Lafutadine and Domperidone in Capsule by High Performance Liquid Chromatography, Global Journal of Pharmacology, 6(2): 60-64.
10. Yogeswari, S., S. Ramalakshmi, R. Neelavathy and J. Muthumary, 2012. Identification and Comparative Studies of Different Volatile Fractions from Monochaetia kansensis by GCMS, Global Journal of Pharmacology, 6(2): 65-71.