Journal of Computer Science 7 (2): 206-215, 2011
ISSN 1549-3636

An Investigation of Using Parallel Genetic Algorithm for Solving the Shortest Path Routing Problem

[1]Salman Yussof, [1]Rina Azlin Razali and [2]Ong Hang See
[1]Department of Systems and Networking,
College of Information Technology,
[2]Department of Electronics and Communication Engineering,
College of Engineering,
Universiti Tenaga National, Jalan IKRAM-Uniten, 43000 Kajang, Malaysia

Abstract: Problem statement: Shortest path routing is the type of routing widely used in computer network nowadays. Even though shortest path routing algorithms are well established, other alternative methods may have their own advantages. One such alternative is to use a GA-based routing algorithm. According to previous researches, GA-based routing algorithm has been found to be more scalable and insensitive to variations in network topologies. However, it is also known that GA-based routing algorithm is not fast enough for real-time computation. **Approach:** To improve the computation time of GA-based routing algorithm, this study proposes a coarse-grained parallel GA routing algorithm for solving the shortest path routing problem. The proposed algorithm is evaluated using simulation where the proposed algorithm is executed on networks with various topologies and sizes. The parallel computation is performed using an MPI cluster. Three different experiments were conducted to identify the best value for the migration rate, the accuracy and execution time with respect to the number of computing nodes and speedup achieved as compared to the serial version of the same algorithm. **Results:** The result of the simulation shows that the best result is achieved for a migration rate around 0.1 and 0.2. The experiments also show that with larger number of computing nodes, accuracy decreases linearly, but computation time decreases exponentially, which justifies the use parallel implementation of GA to improve the speed of GA-based routing algorithm. Finally, the experiments also show that the proposed algorithm is able to achieve a speedup of up to 818.11% on the MPI cluster used to run the simulation. **Conclusion/Recommendations:** We have successfully shown that the performance of GA-based shortest path routing algorithm can be improved by using a coarse-grained parallel GA implementation. Even though in this study the proposed algorithm is executed using an MPI cluster, the algorithm is also applicable to other parallel architecture such as multi-core CPU, multi-processor or GPGPU. A future work would be to evaluate the performance of the proposed algorithm on these other parallel architectures.

Key words: Parallel genetic algorithm, coarse-grained, shortest-path routing, message passing interface, parallel architecture, routing algorithm, Genetic algorithm (GA), parallel computer, Message Passing Interface (MPI), smaller networks, computer network

INTRODUCTION

Routing in a computer network refers to the task of finding a path from a source node to a destination node. Given a particular network, it is very likely that there is more than one path that can be used. The task of a routing algorithm is to find the shortest path. Shortest path routing algorithms such as Dijkstra's algorithm and Bellman-Ford algorithm are commonly used in computer network nowadays (Kurose and Ross, 2010).

Even though shortest path routing algorithms are already well established, there are researchers who are trying to find alternative methods to find shortest paths through a network. These alternative methods commonly employ AI techniques such as genetic algorithm (Munetomo *et al*., 1998; Ahn and Ramakrishna, 1999), neural networks (Liu and Wang, 2009), particle swarm optimization (Mukhef *et al*., 2008; Yusoff *et al*., 2010), ant colony optimization (Zakzouk *et al*., 2010; Guo *et al*., 2010), simulated annealing algorithm (Su and Li, 2009) and A* search algorithm (Panich, 2010).

Corresponding Author: Salman Yussof, Department of Systems and Networking, College of Information Technology, Universiti Tenaga National, Jalan IKRAM-Uniten, 43000 Kajang, Malaysia

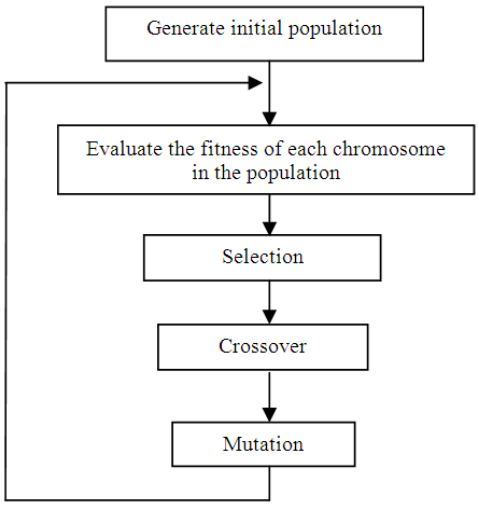

Fig. 1: The general outline of GA

Genetic Algorithm (GA) is a multi-purpose search and optimization algorithm that is inspired by the theory of genetics and natural selection (Goldberg, 1989). The problem to be solved using GA is encoded as a chromosome that consists of several genes. The solution of the problem is represented by a group of chromosomes referred to as a population. During each iteration of the algorithm, the chromosomes in the population will undergo one or more genetic operations such as crossover and mutation. The result of the genetic operations will become the next generations of the solution. This process continues until either the solution is found or a certain termination condition is met. The idea behind GA is to have the chromosomes in the population to slowly converge to an optimal solution. At the same time, the algorithm is supposed to maintain enough diversity so that it can search a large search space. It is the combination of these two characteristics that makes GA a good search and optimization algorithm. The general outline of GA is shown in Fig. 1.

One of the earliest GA-based shortest path routing algorithms is the one proposed by Munetomo *et al*. (1998; 2001). Munetomo proposed a GA-based routing algorithm to generate alternate paths that can be quickly used in the case of link failures. In the proposed algorithm, the algorithm chromosome is encoded as a list of node IDs that are on the path from the source node to the destination node. Since different paths can have different number of nodes, the chromosomes are of variable length. This algorithm employs crossover, mutation and migration genetic operators in generating the next generation of solutions. Ahn and Ramakrishna (1999) also proposed a GA-based routing algorithm for solving the shortest path routing problem. Similar to Munetomo's algorithm, the chromosome in this algorithm consists of a sequence of node IDs that are on the path from source to destination. However, there are several differences in the details of the GA implementation such as in the crossover and mutation operations. Some researchers implemented a hybrid GA algorithm where GA is combined with another algorithm to solve the shortest path routing problem. One example of this is the algorithm proposed by Hamdan and El-Hawary (2002) that combined GA with the Hopfield network. Another example would be an algorithm proposed by Riedl (2002) who combined GA with a local heuristics search.

Ahn and Ramakrishna (1999) has shown that using GA for the shortest path routing problem has several advantages. The first advantage is that GA is insensitive to variations in network topologies with respect to route optimality and convergence speed. The second advantage is that GA-based routing algorithm is scalable in the sense that the real computation size does not increase very much as the network size gets larger. However this literature also pointed out that GA is not fast enough for real-time computation and in order to achieve a really fast computation time in GA, a hardware implementation of GA is required.

Due to the advantages of GA-based routing algorithms, there are researchers who have extended these algorithms to solve the more difficult multi-constrained path routing problem where each network link has more than one parameter the path chosen must fulfill a specific QoS requirement or constraints (Yussof and Ong, 2010; Potti and Chinnasamy, 2011). GA has also been used to solve other types of routing problems such as vehicle routing (Nazif and Lee, 2010), logistics distribution routing (Zaoqiang and Minde, 2009), evacuation route assignment (Li *et al.*, 2010), aircraft route planning (Gao and Zheng, 2010), traffic route choice problem (Li and Zhu, 2010) and travelling salesman problem (Al Rahedi and Atoum, 2009).

In this study, we are proposing a parallel genetic algorithm for the shortest path routing problem. The motivation behind this proposal is that parallel implementation of GA should be able to improve its computation time. The proposed algorithm is implemented on a Message Passing Interface (MPI) cluster.

Parallel genetic algorithm: GA is generally able to find good solutions in reasonable amount of time but as they are applied to harder and bigger problems, there is an increase in the time required to find adequate solutions. As

a consequence, there have been multiple efforts to make GA faster and one of the promising choices is to use parallel implementation (Paz, 2000).

In parallel GA, there are multiple computing nodes. The task of each computing node depends on the type of parallel GA used. There are four major types of parallel GAs which are master-slave GA, coarse-grained GA, fine-grained GA and hierarchical hybrids. In master-slave GA, one computing node will become the master and the other computing nodes will become the slaves. The master node will hold the population and perform most of the GA operations. However, the master can assign one or more computing-intensive tasks to the slaves. This is done by sending one or more chromosomes to the slaves and the master would then wait for the slaves to return their results. In coarse-grained GA, the population is divided among the computing nodes and each computing node executes GA on its own sub-population. To ensure that good solutions can be spread to other nodes, the nodes can occasionally, with certain probability, exchange chromosomes with each other. This exchange is called migration and it involves a node sending a chosen chromosome to other nodes. The other nodes would then replace a chromosome in their population with the one received. Which node is chosen to be migrated or replaced would depend on the migration strategy used. Fine-grained GA has the highest level of parallelism among the four types of parallel GAs. In fine-grained GA, each computing node only has a single chromosome. The computing nodes are normally arranged in a spatial structure where each node can only communicate with several neighboring nodes. The population would be the collection of all the chromosomes in each node. To execute a genetic operation, a computing node will have to interact with its neighbors. Since the neighborhood overlaps, eventually the good traits of a superior individual can spread to the entire population. Fine-grained GA has a large communication overhead due to the high frequency of interactions between neighboring nodes. The final parallel GA type is the hierarchical hybrid which is structured in two levels. At the higher level, the algorithm operates as a coarse-grained GA while at the lower level the algorithm operates as a fine-grained (or master-slave) GA.

For each of the parallel GA type, there are multiple variations proposed by researchers to improve its performance or to suite a particular problem. For example, Golub and Jakobovic (2000) proposed a master-slave GA where the master only creates the population and let the slaves perform the whole evolution process. Tan *et al*. (2002) proposed a coarse-grained GA which has a special computing node assigned to collect the best chromosomes from all the nodes and distribute one or more of the fittest ones to the other nodes. This is done to reduce the delays in propagating the globally fittest chromosome to all the computing nodes. De Toro *et al*. (2002) also proposed a modification to the coarse-grained GA where a master node is assigned to gather the whole population from all the computing nodes once in several generations. The master node would then sort the chromosomes according to some objective function and then distribute them again to the computing nodes.

Parallel GA has been successfully used in various problems such as design optimization (Atiqullah, 2002), transport route planning (Meghanathan and Skelton, 2007), time series forecasting (Eklund, 2003; Ourdighi and Benyettou, 2010), network design (Huang *et al*., 1997) and sorting (Han, 1999).

Proposed parallel genetic algorithm for shortest path routing:

Overview: The proposed algorithm will use coarse-grained parallel GA. The main reason for this choice is due to the use of MPI cluster. In an MPI environment, communication between computers has a large overhead. Therefore, coarse-grained GA would be the most suitable type of parallel GA to be used since it has the lowest communication overhead compared to the other parallel GA types.

In our parallel GA implementation, all the computing nodes will randomly create their own sub-population and each of them will execute GA on its own sub-population. However, one of the computing nodes will be assigned the task to gather results from all the other nodes and then choose the best result (the one that gives the shortest path) to be the output of the coarse-grained parallel GA. This node is called the collector node.

The operations done by the computing nodes are outlined below:

- Randomly initialize the initial sub-population
- Evaluate fitness of each chromosome in the population
- Create the mating pool which consists of all the chromosomes in the current population
- Apply crossover operator several times to create *n* new children for the new sub-population (where *n* is the size of the sub-population). The parents are selected using the selection operator. Crossover is only performed if one or both of the parents have not yet mated

- Apply mutation operator on the chromosomes in the mating pool. Each chromosome has a certain probability to be mutated
- If migration rate is higher than 0, decide whether migration is to be performed in this iteration. If yes, choose the best chromosome and send it to the other computing nodes. At the same time, check whether there is any chromosome migrated from the other computing nodes. For each chromosome received, replace the worst chromosome in the population with the migrated chromosome if the migrated chromosome has better fitness.
- Repeat step 2 until the sub-population converges or the maximum number of iterations has been achieved. The value for maximum iteration used in all the experiments is 100
- Send the best chromosome to the collector node

The operations done by the collector node are outlined below:

- Receive the best chromosomes from each of the computing nodes
- Choose the best chromosome from the ones received. This is presented as the shortest path found by the algorithm

Genetic encoding: A communication network can be modeled as a directed graph G (N, E), where N is the set of nodes representing routers and E is the set of edges connecting the links that connect between the routers (Kurose and Ross, 2010). Each edge (i,j) is associated with an integer representing the cost of sending data from node i to node j and vice versa.

In the proposed algorithm, each chromosome is encoded as a series of node IDs 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.

Initial sub-population: In the beginning, the sub-population is filled with chromosomes that represent random paths. Even though the paths are random, they are supposed to be valid paths, where the chromosomes consist of a sequence of nodes that are in the path from sender to receiver. The number of chromosomes generated for each sub-population, S_n, depends on the total population size and the number of computing nodes, as depicted in Eq. 1:

$$S_n = \frac{P}{N} \quad (1)$$

where, S_n represents the sub-population size of the nth node, P represents the total population size and N represents the total number of computing nodes.

The algorithm used to generate the random paths is as follows:

- Start from the source node
- Randomly choose, with equal probability, one of the nodes that are connected to the current node
- If the chosen node has not been visited before, mark that node as the next node in the path. Otherwise, find another node
- If all the neighboring nodes have been visited, go back to step 1
- Otherwise, repeat step 2 by using the next node as the current node
- Do this until the destination node is found

Fitness function: Each chromosome in the population is associated with a fitness value that is calculated using a fitness function. This value indicates how good the solution is for a particular chromosome. This information is then used to pick the chromosomes that will contribute to the formation of the next generation of solution. The fitness function used in the proposed algorithm is defined as follows:

$$f_i = \frac{1}{c_i} \quad (2)$$

where, f_i represents the fitness value of the i^{th} chromosome and c_i represents the total cost of the path represented by the i^{th} chromosome. This would give a higher fitness value for shorter paths.

Selection: Selection is used to choose the parent chromosomes for the crossover operation. The selection scheme used in the algorithm is the pairwise tournament selection with tournament size, s = 2. 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 at least one common node other than the source and destination nodes. If more than one common node exists, one of them will be randomly chosen with equal probability. The chosen 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 node and destination node 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 vice versa. 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, p_m. For all the experiments, the value for p_m is set to 0.05. For each chromosome that is chosen to be mutated, a mutation point will be chosen randomly, with equal probability; among the intermediate nodes in the path from sender to receiver (i.e., the sending and receiving node cannot be chosen as the mutation point). Once the mutation point is chosen, the chromosome will be changed starting from the node after the mutation point and onwards. For example, assume that the following chromosome has been chosen to be mutated:

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

where, A and Z are the sending node and the receiving node respectively. Assume also that the node G has been chosen 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 i^{th} new node in the path. The new path is generated randomly; the same way as the paths in the initial population is generated.

Migration: Migration is a genetic operation commonly used in coarse-grained parallel GA implementation (Goldberg, 1989). In coarse-grained parallel GA, each computing node has its own sub-population that evolves independently and in isolation. As compared to the serial GA, this would result in low diversity of the population because different sub-populations do not interact with each other. Migration is an operation that can be used to increase the sub-population diversity by having the computing nodes to share their results with each other. Migration involves having each computing node sending one of its chromosomes to the other nodes. At the same time, each computing node will receive migrated chromosomes from the other nodes. The received chromosome can replace one of the chromosomes currently in the sub-population. The migration rate is controlled by the migration probability, p_{mg}.

There are several different migration strategies, as discussed by Goldberg (1989). Each computing node can either send its best chromosome or a random chromosome from its sub-population. The receiving node, on the other hand, can either choose to replace its worst chromosome or just any random chromosome in its sub-population. For the proposed algorithm, the migration strategy used is to send the best chromosome and replace the worst chromosome.

MATERIALS AND METHOD

The proposed algorithm is implemented as a C++ program. The program is run on an MPI cluster which has six machines. Each machine has a dual-core processor. A computing node is associated with a single processing core. Therefore, up to 12 computing nodes can be run on this parallel computer.

The objective of this experiment is to measure the performance of the proposed algorithms with respect to accuracy and computation time. Accuracy measures the percentage of the shortest paths returned by the algorithm that are actually shortest paths (as obtained from Dijkstra's algorithm). Computation time measures the execution time taken by the algorithm to obtain all the results from the beginning to the end of the simulation. The performance of the algorithm is compared with the non-parallel version of the same algorithm.

There are two types of network used in the simulation, the n x n mesh network and the Waxman network (Waxman, 1988). The Waxman network is

actually a random graph where the existence of link between two nodes, i and j, is defined by the following probability:

$$p_{ij} = \alpha\exp(-(\frac{d_{i,j}}{\beta L})), 0 < \alpha, \beta < 1 \quad (3)$$

where, $d_{i,j}$ is the distance between the two nodes and L is the maximum inter-nodal distance in the topology. A larger value of α would generate a graph with higher density and a smaller value of β increases the density of short edges relative to longer ones. In all the experiments, the values for both α and β are set to 0.2 and 0.1 respectively. However, to avoid having disconnected nodes, each node must be connected to at least one other node. The network topologies used are 10×10 mesh network 15×15 mesh network, 100-node Waxman network and 225-node Waxman network. Each link in the network is given a randomly generated cost value, $c_k(i,j) \sim$ uniform[1,20].

The result reported in the next section is averaged over 50 runs. For each run, a new network with a new set of link metrics is randomly generated using different seeds. For the Waxman network, this also means that a different network topology is generated on each run. In each run, a total of 1000 source-destination pairs are randomly chosen and the shortest path for each of them is computed.

RESULTS

Three different experiments are conducted to evaluate the proposed algorithm. The first experiment aims to find the most suitable migration rate, p_{mg}, to be used. The second experiment evaluates the accuracy and execution of the algorithm with respect to the number of computing nodes. The third experiment evaluates the speedup achieved by the parallel GA routing algorithm as compared to a serial version of the same algorithm.

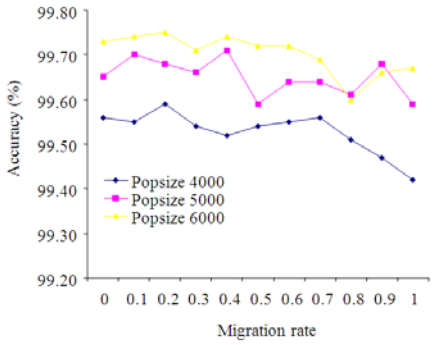

Fig. 2: Accuracy for 10×10 mesh network

The result of the first experiment is presented in Fig. 2 until Fig. 5. These figures present the accuracy result with respect to the migration rate for the four network topologies used, where Fig. 2 and Fig. 3 show the result for t hen x n mesh networks while Fig 4 and Fig. 5 show the results for the Waxman networks. For the mesh networks, it seems that having migration does not necessarily result in better accuracy. Having no migration (p_{mg} = 0) seems to be better than having a migration rate that is too high.

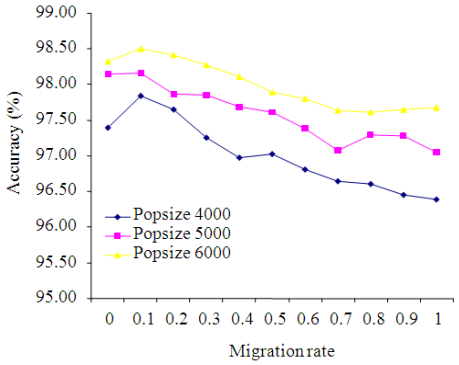

Fig. 3: Accuracy for 15×15 mesh network

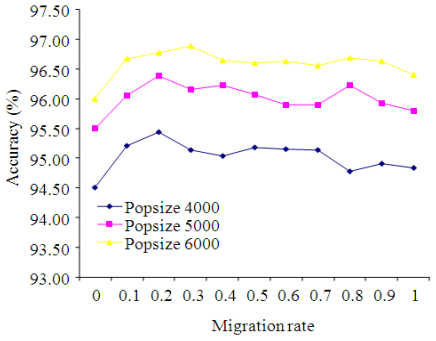

Fig. 4: Accuracy for 100-node Waxman network

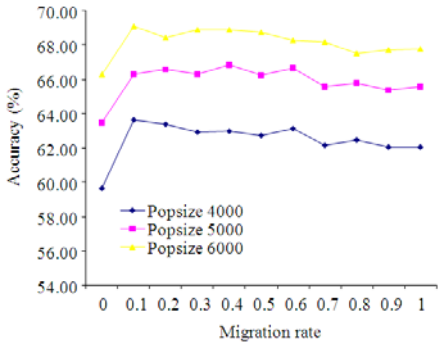

Fig. 5: Accuracy for 225-node Waxman network

However, the best result is achieved by using a low migration rate of 0.1 or 0.2. For the Waxman network, having a non-zero migration rate is definitely better than having no migration at all. Again, the best result is achieved with a low migration rate of 0.1 or 0.2. Based on the result of this experiment, subsequent experiments are performed using $p_{mg} = 0.1$.

The result of the second experiment is depicted in Fig. 6 until Fig. 9. Fig. 6 and Fig. 7 show the accuracy of the proposed algorithm for the n x n mesh network and the Waxman network respectively. It is obvious that as the number of computing node increases, the accuracy decreases linearly. This is because as the number of computing node increases, each computing node would have smaller sub-population size. In GA, smaller population size would result in lower performance. However, the benefit of PGA is apparent when comparing the computation time as depicted in Fig. 8 and Fig. 9. For both types of network, as the number of computing node increases, the computation time decreases exponentially. However, it is also observed that for small sub-population size, having too many computing nodes may eventually increase the computation time again. This is because with small sub-population size, the parallelism is not fully exploited due to the computing nodes are not doing enough work and the communication overhead would then cause the computation time to increase. This experiment also shows that having a higher total population size would increase the accuracy. This effect is consistent regardless of the network topology, network size or the number of computing nodes. Having a higher population size would also increase the computation time. However, with larger number of computing nodes, the increase in computation time becomes less apparent.

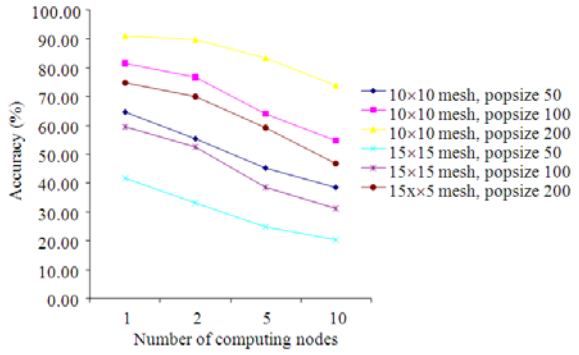

Fig. 6: Accuracy for n×n mesh networks

The result of the third experiment is depicted in Table 1. This table shows the percentage of computation time required by the proposed PGA (T_{PGA}) and the computation time of its serial counterpart (T_{GA}) to get relatively similar accuracy. Based on the result, the coarse-grained parallel GA algorithm is able to achieve a speedup from 546.05% – 818.11% relative to the serial GA implementation of the same algorithm.

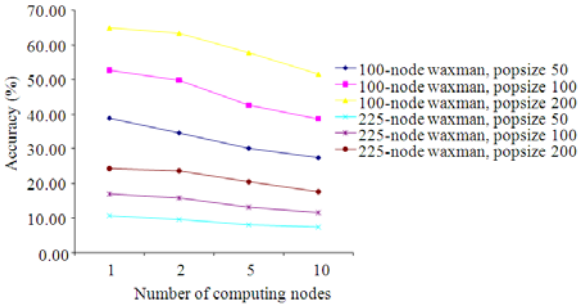

Fig. 7: Accuracy for waxman networks

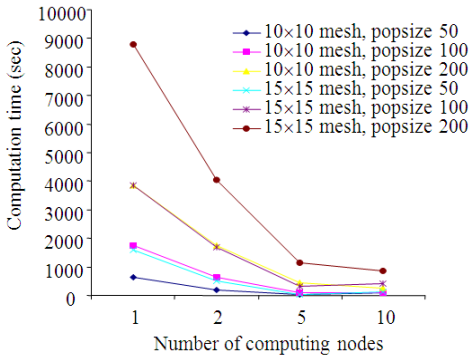

Fig. 8: Computation time for n×n mesh networks

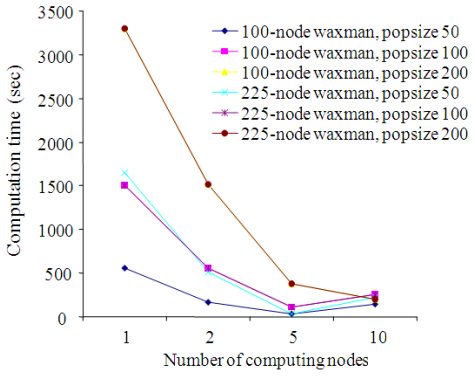

Fig. 9: Computation time for waxman networks

Table 1: Speed up achieved by the proposed algorithm

Network topology	T_{PGA} (sec)	T_{GA} (sec)	T_{GA}/T_{PGA} * 100 (%)
10×10 mesh	2872.73	19440.7	676.73
15×15 mesh	5666.97	30945.0	546.05
100-node Waxman	2765.28	22623.1	818.11
225-node Waxman	6201.05	35739.6	576.34

DISCUSSION

Based on the results obtained from these experiments, it can be concluded that the parallel GA implementation of the GA-based routing algorithm can help to increase the speed of the algorithm while at the same time maintaining a high quality result. This conclusion is justified based on the observation where with larger number of computing nodes, accuracy decreases linearly, but computation time decreases exponentially. However, similar to other parallel algorithm implementations, the parallelism will only provide a significant advantage when the problem is big enough. Otherwise, the communication overhead between computing nodes will bring down the performance.

With respect to the shortest path routing problem, the problem will be big enough when the number of nodes and the number of paths to be evaluated is large. For the GA-based shortest path routing algorithm, the parallel implementation will also provide a significant advantage when then population size is large. In GA, large population size leads to better results but at the same time increases the computation time. Parallel implementation of GA can really help to reduce computation time while maintaining the quality of the results.

CONCLUSION

This study proposed a coarse-grained parallel genetic algorithm for solving the shortest path routing problem. A series of experiments were conducted to evaluate various aspects of the algorithm. Based on the experiments conducted, it was determined that the best result is achieved with a low migration rate, p_{mg}, of around 0.1 and 0.2. The experiments also show that with larger number of computing nodes, accuracy decreases linearly, but computation time decreases exponentially. This exponential decrease in computation time as compared to linear decrease in accuracy justifies the use parallel implementation of GA to improve the performance of GA-based routing algorithm. Finally, the experiments also show that the proposed algorithm is able to achieve a speedup of up to 818.11% on the 12-node MPI cluster. Even though in this study the proposed algorithm is executed using an MPI cluster, the algorithm is also applicable to other parallel architecture such as multi-core CPU, multi-processor or GPGPU. A future work would be to evaluate the performance of the proposed algorithm on these other parallel architectures.

REFERENCES

Ahn, C.W. and R.S. Ramakrishna, 1999. A genetic algorithm for shortest path routing problem and the sizing of populations. IEEE Trans. Evolutionary Comput., 6: 566-579. DOI: 10.1109/TEVC.2002.804323

Al Rahedi, N.T. and J. Atoum, 2009. Solving the traveling salesman problem using new operators in genetic algorithms. Am. J. Applied Sci., 6: 1586-1590. DOI: 10.3844/ajassp.2009.1586.1590

Atiqullah, M.M., 2002. Problem independent parallel genetic algorithm for design optimization. Proceeding of the International Symposium on Parallel and Distributed Processing, (ISPDP'02), Fort Lauderdale, Florida, pp: 204-211. DOI: 10.1109/IPDPS.2002.1016614

De Toro, F., J. Ortega, J. Fernandez and A. Diaz, 2002. PSFGA: a parallel genetic algorithm for multiobjective optimization. Proceeding of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, Jan. 09-11, Canary Islands, Spain, pp: 384-391. DOI: 10.1109/EMPDP.2002.994315

Eklund, S.E., 2003. Time series forecasting using massively parallel genetic programming. Proceeding of the International Symposium on Parallel and Distributed Processing, Apr. 22-26, Dalarna University, Sweden, pp: 5-5. DOI: 10.1109/IPDPS.2003.1213272

Gao, Y. and T. Zheng, 2010. Chaos genetic algorithm for aircraft route planning problem. Proceeding of the Second Global Congress on Intelligent Systems, Dec. 16-17, Wuhan, Hubei, China, pp: 280-284. DOI: 10.1109/GCIS.2010.206

Goldberg, D.E., 1989. Genetic Algorithm in Search, Optimization and Machine Learning, 1st Edn., Addison Wesley, United States, ISBN: 0201157675, pp: 412.

Golub, M. and D. Jakobovic, 2000. A new model of global parallel genetic algorithm. Proceeding of the 22nd International Conference on Information Technology Interfaces, June 13-16, Zagreb Univ., Croatia, pp: 363-368. DOI: 10.1109/ITI.2000.915963

Guo, Y., Z. Qin and Y. Chang, 2010. A novel hybrid algorithm for the dynamic shortest path problem. Proceeding of the 6th International Conference on Natural Computation, Aug. 10-12, Yantai, Shandong, pp: 2545-2550. DOI: 10.1109/ICNC.2010.5583241

Hamdan, M. and M.E. El-Hawary, 2002. Hopfield-genetic approach for solving the routing problem in computer networks. Proceeding of the Canadian Conference on Electrical and Computer Engineering, (CCECE'02), Dalhousie Univ., Halifax, NS., pp: 823-827. DOI: 10.1109/CCECE.2002.1013048

Han, M.M., 1999. Applying parallel genetic algorithm to sorting problem. Proceeding of the IEEE International Conference on Fuzzy Systems, Aug. 22-25, Seoul , South Korea, pp: 1796-1801. DOI: 10.1109/FUZZY.1999.790180

Huang, R., J. Ma, T.L. Kunii and E. Tsuboi, 1997. Parallel genetic algorithms for communication network design. Proceeding of the 2nd Aizu International Symposium on Parallel Algorithms/Architecture Synthesis, Mar. 17-21, Aizu-Wakamatsu , Japan, pp: 370-377. DOI: 10.1109/AISPAS.1997.581701

Kurose, J.F. and K.W. Ross, 2010. Computer Networking: A Top-down Approach. 5th Edn., Pearson Education, CA., ISBN0131365487, pp: 888.

Li, J. and M. Zhu, 2010. An implementation of genetic algorithm in Matlab: solution to the route choice problem in the urban traffic network. Proceeding of International Conference on Computational and Information Sciences, Dec. 17-19, Chengdu, China, pp: 638-641. DOI: 10.1109/ICCIS.2010.160

Li, Q., Z. Fang, Q. Li and X. Zong, 2010. Multiobjective evacuation route assignment model based on genetic algorithm. Proceeding of the 18th International Conference on Geomatics, June 18-20, Beijing, pp: 1-5. DOI: 10.1109/GEOINFORMATICS.2010.5567485

Liu, W. and L. Wang, 2009. Solving the shortest path routing problem using noisy hopfield neural networks. Proceeding of the International Conference on Communications and Mobile Computing, Jan. 6-8, Nanyang Technol. Univ., Singapore, pp: 299-302. DOI: 10.1109/CMC.2009.366

Meghanathan, N. and G.W. Skelton, 2007. Intelligent transport route planning using parallel genetic algorithms and MPI in high performance computing cluster. Proceeding of the International Conference on Advanced Computing and Communications, Dec. 18-21, Guwahati, Assam, pp: 578-583. DOI: 10.1109/ADCOM.2007.76

Mukhef, H.A., E.M. Farhan and M.R. Jassim, 2008. Generalized shortest path problem in uncertain environment based on PSO. J. Comput. Sci., 4: 349-352. DOI: 10.3844/jcssp.2008.349.352

Munetomo, M., N. Yamaguchi, K. Akama and Y. Sato, 2001. Empirical investigations on the genetic adaptive routing algorithm in the Internet. Proceeding of the 2001 Congress on Evolutionary Computation, May 27-30, Seoul , South Korea, pp: 1236-1243. DOI: 10.1109/CEC.2001.934332

Munetomo, M., Y. Takai and Y. Sato, 1998. A migration scheme for the genetic adaptive routing algorithm. Proceeding of the IEEE International Conference on Systems, Man and Cybernetics, Oct. 11-14, San Diego, CA , USA., pp: 2774-2779. DOI: 10.1109/ICSMC.1998.725081

Nazif, H. and L.S. Lee, 2010. Optimized crossover genetic algorithm for vehicle routing problem with time windows. Am. J. Applied Sci., 7: 95-101. DOI: 10.3844/ajassp.2010.95.101

Ourdighi, A. and A. Benyettou, 2010. An adaptive time-delay neural network training using parallel genetic algorithms in time-series prediction and classification. J. Applied Sci., 10: 2115-2120. DOI: 10.3923/jas.2010.2115.2120

Panich, S., 2010. The shortest path with intelligent algorithm. J. Math. Stat., 6: 276-278. DOI: 10.3844/jmssp.2010.276.278

Paz, E.C., 2000. Efficient and Accurate Parallel Genetic Algorithms. 1st Edn., Springer, India, ISBN-10: 0792372212, pp: 162.

Potti, S. and C. Chinnasamy, 2011. Strength pareto evolutionary algorithm based multi-objective optimization for shortest path routing problem in computer networks. J. Comput. Sci., 7: 17-26. DOI: 10.3844/jcssp.2011.17.26

Riedl, A., 2002. A hybrid genetic algorithm for routing optimization in IP networks utilizing bandwidth and delay metrics. Proceeding of the IEEE Workshop on IP Operations and Management, (WIPOM'02), Munich Univ. of Technol., Germany, pp: 166-170. DOI: 10.1109/IPOM.2002.1045774

Su, J. and A. Li, 2009. Approach to the shortest path with fuzzy constraints by simulated annealing algorithm. Proceeding of the WRI Global Congress on Intelligent Systems, May 19-21, Xiamen, pp: 516-520. DOI: 10.1109/GCIS.2009.20

Tan, L., D. Taniar and K.A. Smith, 2002. A new parallel genetic algorithm. Proceeding of the International Symposium on Parallel Architectures, Algorithms and Networks, May 22-24, Makati City, Metro Manila , Philippines, pp: 284-89. DOI: 10.1109/ISPAN.2002.1004301

Waxman, B.M., 1988. Routing of multipoint connections. IEEE J. Selected Areas Communi., 6: 1617-1622. DOI: 10.1109/49.12889

Yusoff, M., J. Ariffin and A. Mohamed, 2010. A discrete particle swarm optimization with random selection solution for the shortest path problem. Proceeding of the International Conference of Soft Computing and Pattern Recognition, Dec. 7-10, Paris, pp: 133-138. DOI: 10.1109/SOCPAR.2010.5685867

Yussof, S. and H.S. Ong, 2010. A robust GA-based QoS routing algorithm for solving multi-constrained path problem. J. Comput., 5: 1322-1334. DOI: 10.4304/jcp.5.9.1322-1334

Zakzouk, A.A.A., H.M. Zaher and R.A.Z. El-Deen, 2010. An ant colony optimization approach for solving shortest path problem with fuzzy constraints. Proceeding of the 7th International Conference on Informatics and Systems, Mar. 28-30, Cairo, pp: 1-8. ISBN: 978-1-4244-5828-8

Zaoqiang, C. and S. Minde, 2009. The research of the logistics distribution routing optimization based on immune genetic algorithm. Proceeding of the International Conference on Artificial Intelligence and Computational Intelligence, Nov. 7-8, Shanghai, pp: 449-452. DOI: 10.1109/AICI.2009.423