Communications and Network, 2012, 4, 93-100
http://dx.doi.org/10.4236/cn.2012.42013 Published Online May 2012 (http://www.SciRP.org/journal/cn)

An Investigation on the Effect of Migration Strategy on Parallel GA-Based Shortest Path Routing Algorithm

Salman Yussof*, Rina Azlin Razali
Department of Systems and Networking, College of Information Technology,
Universiti Tenaga Nasional, Kajang, Malaysia
Email: *salman@uniten.edu.my

Received January 20, 2012; revised February 18, 2012; accepted March 15, 2012

ABSTRACT

Genetic algorithm (GA) is one of the alternative approaches for solving the shortest path routing problem. In previous work, we have developed a coarse-grained parallel GA-based shortest path routing algorithm. With parallel GA, there is a GA operator called migration, where a chromosome is taken from one sub-population to replace a chromosome in another sub-population. Which chromosome to be taken and replaced is subjected to the migration strategy used. There are four different migration strategies that can be employed: best replace worst, best replace random, random replace worst, and random replace random. In this paper, we are going to evaluate the effect of different migration strategies on the parallel GA-based routing algorithm that has been developed in the previous work. Theoretically, the migration strategy best replace worst should perform better than the other strategies. However, result from simulation shows that even though the migration strategy best replace worst performs better most of the time, there are situations when one of the other strategies can perform just as well, or sometimes better.

Keywords: Parallel Genetic Algorithm; Shortest Path Routing; Migration Strategy

1. Introduction

Routing in a computer network refers to the task of finding a path from the source node to the destination node. Given a large network, it is very likely that there is more than one path for each source-destination pair. The task of a routing algorithm is to find the least-cost path among all the paths available for a particular source-destination pair. There are two types of routing algorithm used in the Internet nowadays, which are the link-state routing algorithm and the distance-vector routing algorithm [1].

Even though the link-state and distance vector routing algorithms are well-established in the Internet, the quest for shortest-path routing algorithm is far from over. In recent years, there are many researchers who came up with new shortest path routing algorithms that are based on nature-inspired optimization techniques such as genetic algorithm [2,3], neural networks [4], particle swarm optimization [5,6] and ant colony optimization [7]. These nature-inspired algorithms have several advantages over the traditional link-state or distance-vector routing algorithms. For example, a shortest-path routing algorithm based on genetic algorithm (GA) has the advantage of being more scalable and is insensitive to variations in network topologies with respect to route optimality [3]. Another work by [8] also shows that a GA-based routing algorithm is more robust in an environment where the network parameters can easily change. These advantages make these nature-inspired algorithms attractive to be used in wireless and mobile environment such as a mobile ad-hoc network (MANET).

However, these algorithms do have their disadvantages. For example, a GA-based routing algorithm may not be fast enough for real-time computation [3]. In previous work, we have proposed a shortest-path routing algorithm based on coarse-grained parallel genetic algorithm (PGA) with the goal to improve its performance [9]. Through simulation, it has been shown that the use of PGA is able to improve the performance of GA-based routing algorithm as long as the network size and the number of population are large enough.

The performance of a GA-based shortest path routing algorithm can vary depending on the choice of GA parameters used in the algorithm. There are many parameters in GA that can be fine-tuned to get an optimum result such as the genetic encoding, fitness function, population size, maximum iteration, the selection scheme, crossover rate and mutation rate. With coarse-grained PGA, there is another GA operation involved called migration. In implementing migration, there are two GA parameters that need to be considered. These two para-

*Corresponding author.

meters are the migration rate and the migration strategy. Migration rate specifies how often migration is to be performed, while migration strategy defines how the chromosomes involved in the migration operation are chosen.

In this paper we are studying the effect of various migration strategies on the parallel GA-based shortest path routing algorithm proposed in [9]. There are not many researchers who pay attention to migration strategies, even though they agree that improvement can be achieved if a different migration strategy is applied to their algorithm [10]. One researcher who has done a study specific to migration strategy is Cao Yijia [11]. Cao studied to effect of migration strategies in a parallel GA algorithm developed to solve the economic dispatch problem. Even though there is no concrete conclusion, it does provide an idea on the difference in results when using different migration strategies.

The rest of the paper is organized as follows. In Section 2, the overview of GA and PGA is given. The migration operation and the migration strategies are also described in this section. In Section 3, the parallel GA-based shortest path routing algorithm proposed in [9] is described. Section 4 presents the experiment performed and its result. The paper is then concluded in Section 5.

2. Parallel Genetic Algorithm

2.1. Introduction to Genetic Algorithm

GA is a multi-purpose search and optimization algorithm that is inspired by the theory of genetics and natural selection [12]. 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 **Figure 1**.

2.2. Introduction to Parallel Genetic Algorithm

GA is generally able to find good solutions in reasonable amount of time. However, 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

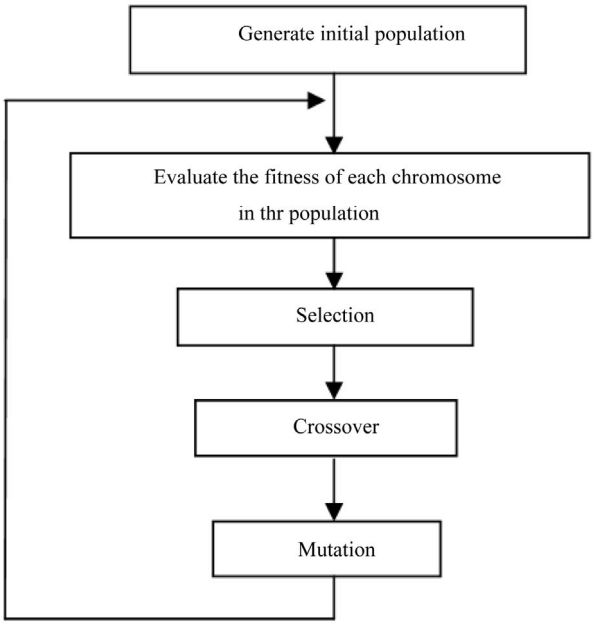

Figure 1. General outline of GA.

the promising options is to use parallel implementation [13]. In PGA, 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 PGAs which are master-slave GA, coarse-grained GA, fine-grained GA and hierarchical hybrids.

In master-slave PGA, 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 PGA, 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 PGA has the highest level of parallelism among the four types of PGAs. In fine-grained PGA, 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 exe-

cute 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 type of PGA is the hierarchical hybrid which is structured in two levels. For example, at the higher level, the algorithm operates as a coarse-grained GA while at the lower level the algorithm operates as a fine-grained GA. A hierarchical PGA combines the benefits of its components and it has better potential than any of the single implementation of the algorithm alone [13].

2.3. Migration Operation

Migration is a genetic operation commonly used in coarse-grained PGA [13]. In coarse-grained PGA, 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. It is commonly performed as the last operation in each iteration. 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 frequency of execution for the migration operation is called the migration rate.

To implement migration, a migration strategy must be chosen. Migration strategy defines which chromosome should be chosen to be migrated from the source sub-population and which chromosome should be replaced by the migrating chromosome in the receiving sub-population [13]. The source sub-population can choose to send the best chromosome or a random chromosome. On the other hand, the receiving sub-population can choose to replace the worst chromosome or a random chromosome. Based on this, there are a total of four possible migration strategies: best replace worst, best replace random, random replace worst, and random replace random. Theoretically, the strategy best replace worst seems to be the best because it allows the distribution of high-fitness chromosomes to replace low-fitness chromosomes. In fact, this is the migration strategy used in many PGA implementations.

3. Parallel GA-Based Shortest Path Routing Algorithm

In [9], we have proposed a coarse-grained PGA for the shortest path routing problem. In the proposed algorithm, the computation is executed by a group of computing nodes. Each computing node will randomly create its own sub-population, and execute GA on its sub-population. The GA computation will be executed until the sub-population converges or the number of maximum iteration has been achieved. Each node will then pass its best result to a special node called the collector node. The collector node will then choose the best result from all the results received to become the output of the algorithm.

The pseudocode below outlines the operation of each computing node.

Randomly initialize the sub-population
While the population has not converged and the maximum number of iteration has not been reached
Evaluate the fitness of each chromosome in the sub-population
Create the mating pool which consists of all the chromosomes in the current sub-population
Perform crossover operation
Perform mutation operation
Perform migration operation
End_while
Send the best chromosome to the collector node

The collector node has the special task of collecting the best chromosome from all the computing nodes and choosing the best one as the output of the algorithm. This task is outlined in the pseudocode below.

While the best chromosomes from all computing nodes have not been received
Receive chromosomes from computing nodes
End_while
Find the best chromosome among the received chromosomes
Present the chromosome as the shortest path found by the algorithm

3.1. 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 [1]. 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 chro-

mosome contains a loop and in network routing, any loop should be eliminated.

3.2. 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 Equation (1):

$$S_n = \frac{P}{N} \tag{1}$$

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 given in the pseudocode below.

Start from the source node
While the destination node is not reached
Randomly choose a neighboring node
If the chosen node is not yet visited
  Mark the node as the next node in the path
  Move to this node
  Continue
End_if
If all the neighboring nodes have been visited
  Move back to the previous node
End_if
End_while

3.3. 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:

$$f_i = \frac{1}{c_i} \tag{2}$$

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.

3.4. 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.

3.5. Crossover

Crossover is performed on the two parent chromosomes selected using the selection scheme described above. Crossover is only performed if one or both chromosomes chosen have not yet mated. 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. For a population of size *n*, the crossover operation must be performed *n*/2 times to generate *n* child chromosomes.

3.6. 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.

3.7. Migration

Migration is performed by choosing a chromosome from the local sub-population and then sends it to the other computing nodes. At the same time, the node should also check whether there is any chromosome migrated from the other computing nodes. If a migrated chromosome is received, it will replace one of the chromosomes in the local sub-population. The rule to determine which chromosome should be chosen to be migrated or replaced depends on the migration strategy used.

4. Experiment and Analysis

As mentioned in Section 2.3, there are four different types of migration strategy which are best replace worst, best replace random, random replace worst, and random replace random. To evaluate the performance of the four migration strategies in the proposed parallel GA-based shortest path routing algorithm, an experiment has been conducted. The performance metric used to measure the performance is accuracy, where accuracy is defined as the percentage of the shortest paths returned by the algorithm that are actually shortest paths (as obtained from Dijkstra's algorithm). A higher accuracy would indicate that the migration strategy used is better.

4.1. Experiment Setup

The algorithm described in the previous section has been implemented as a C++ program which runs on an MPI cluster. There are two types of network used in the simulation, the $n \times n$ mesh network and the Waxman network [14]. The Waxman network is a random graph where the existence of link between two nodes, i and j, is defined by the following probability:

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

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 population size used in this experiment is 5000, and this population is evenly distributed among nine computing nodes. The value for the migration rate is 0.1.

The result reported here 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.

4.2. Results and Discussion

The results of the experiment are presented in **Figures 2-5**, where each figure presents the result for a particular network topology. As expected, the best replace worst strategy tends to give relatively higher accuracy compared to other migration strategies. However, this strategy is not necessarily dominant. It does give the best result in 10 × 10 mesh network (**Figure 2**), 100-node Waxman network (**Figure 4**) and 255-node Waxman network (**Figure 5**). However, in 15 × 15 mesh network (**Figure 3**) the result of this strategy comes second after the random replace random strategy. In fact, in all network topologies used in this experiment, there is one or more of the other migration strategies that perform almost as good as the best replace worst strategy.

Table 1 shows the difference in accuracy between the best and the worst result for each network topology. It also shows the standard deviation between the results of the four migration strategies for each network topology. Based on this, it can be seen that even though there are differences in performance between the migration strategies, the differences is very low. The difference in accuracy between the best and the worst strategy is only between 0.11% and 0.31%. This conclusion can also be drawn from the standard deviation where the standard deviation of the four migration strategies is very low for all network topologies used in this experiment.

The small difference between the four migration strategies can be explained by the fact that in GA, choosing a chromosome with very high fitness and choosing a chromosome with an average fitness both has its own benefits. The issue is similar to the choice of selection scheme in the selection process. Choosing a chromosome with very high fitness increases the selection pressure and this causes the algorithm to converge faster. However, the algorithm may converge to a local optimum. Choosing a chromosome with an average fitness decreases the selection pressure and increases the diversity of the solutions in the population. This allows the algorithm to explore a

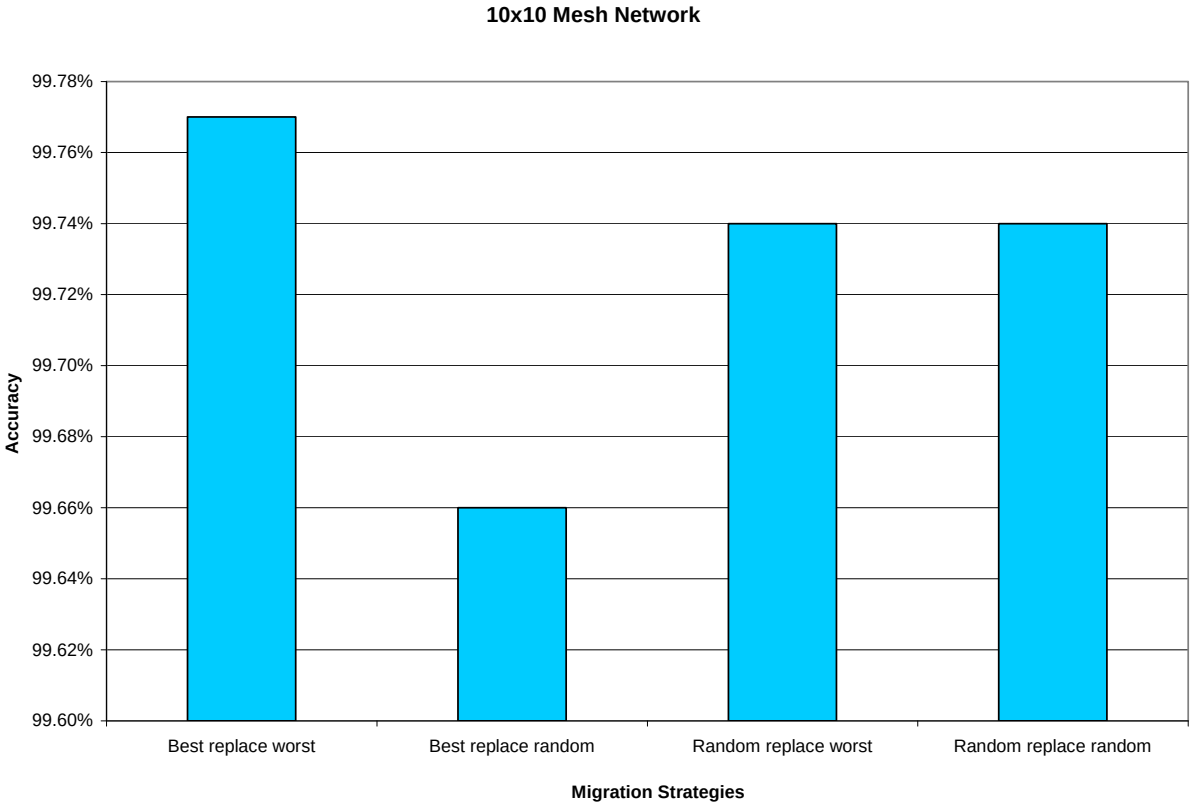

Figure 2. Performance of the four migration strategies in 10 × 10 mesh network.

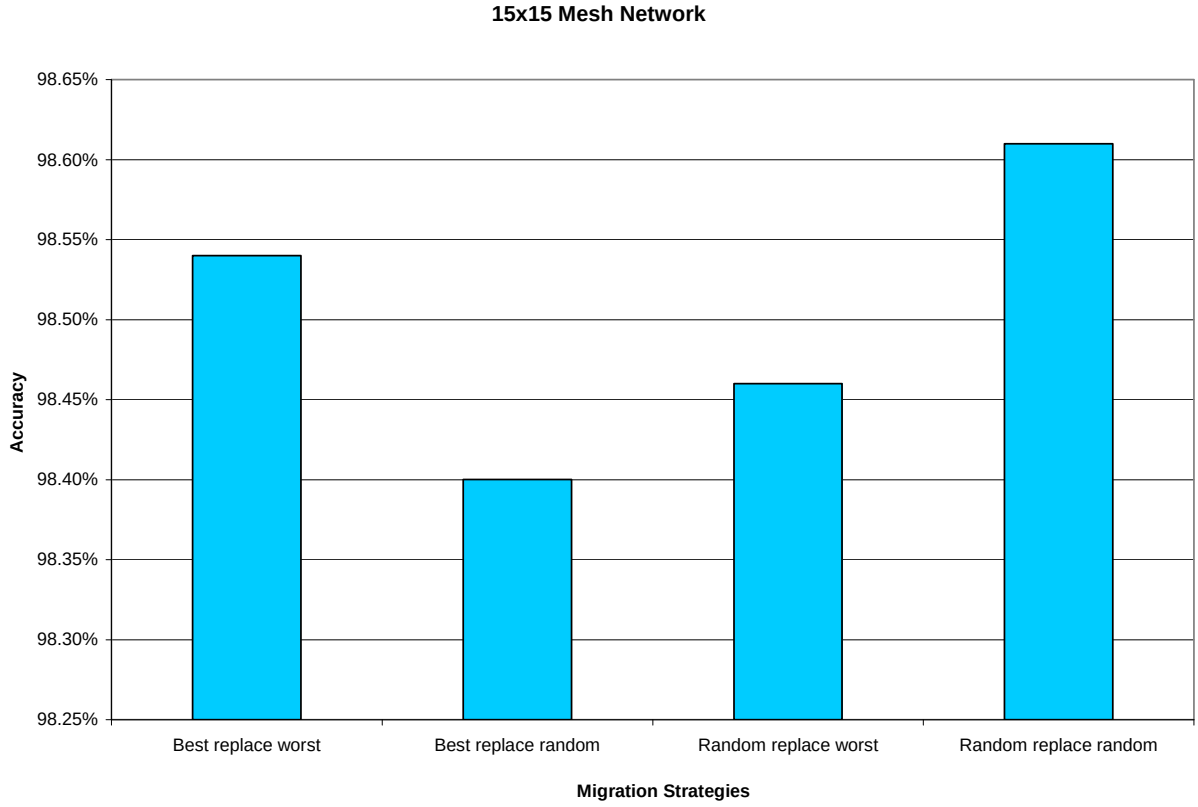

Figure 3. Performance of the four migration strategies in 15 × 15 mesh network.

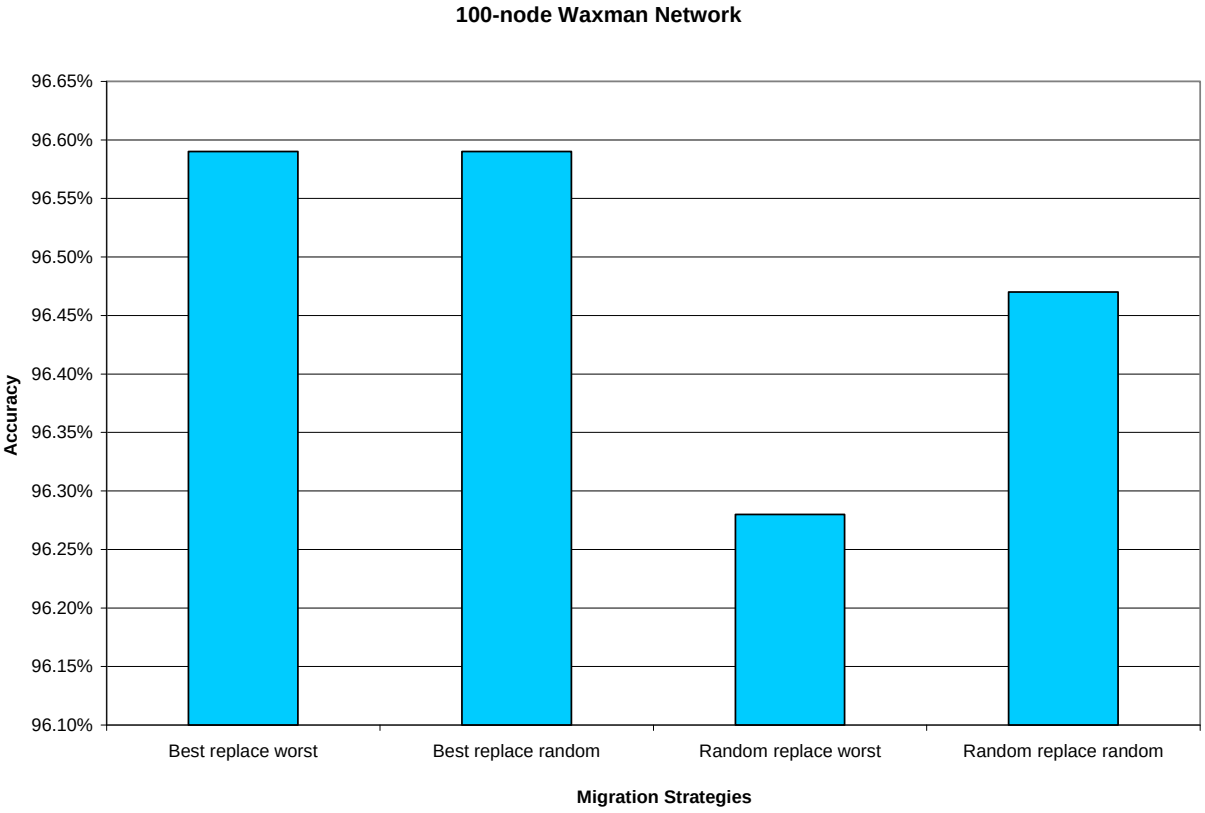

Figure 4. Performance of the four migration strategies in 100-node Waxman network.

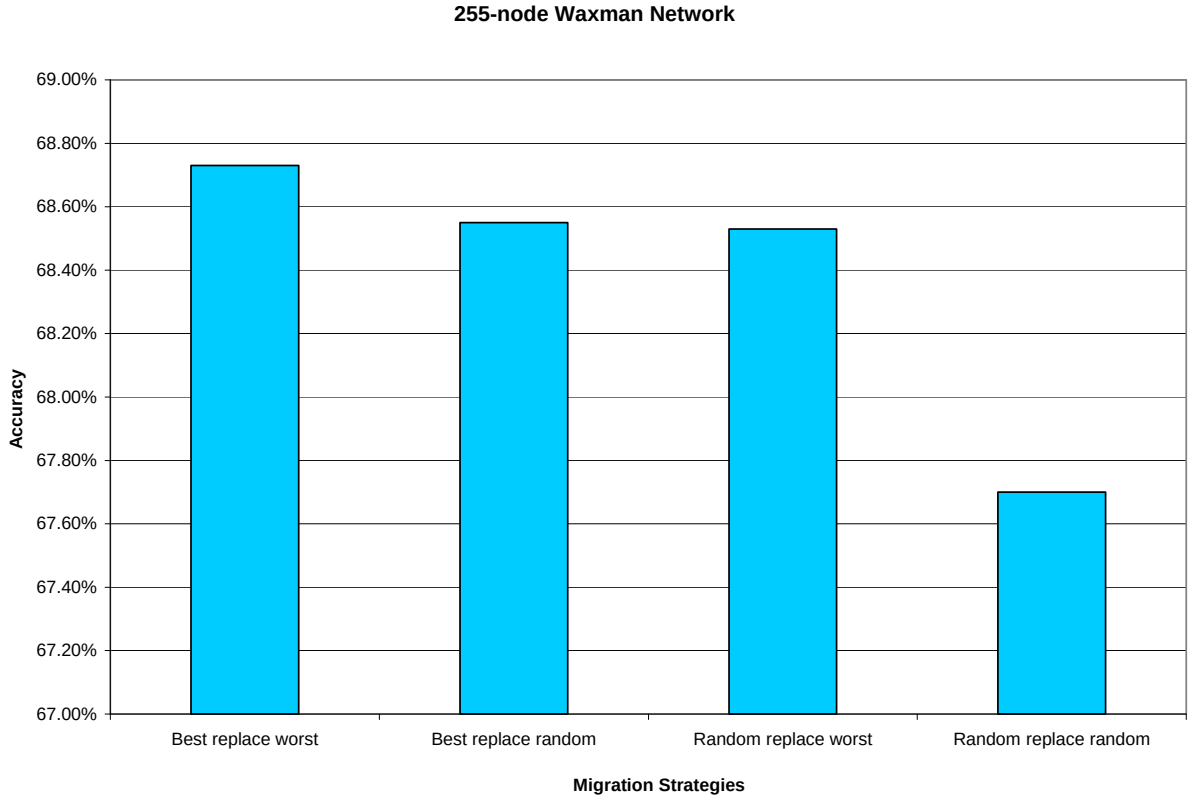

Figure 5. Performance of the four migration strategies in 225-node Waxman network.

Table 1. Statistics of experimental results for each network topology.

Network topology	Difference between the best and the worst result	Standard deviation
10 × 10 mesh	0.11%	0.000471699
15 × 15 mesh	0.21%	0.000917878
100-node Waxman	0.31%	0.001463728
255-node Waxman	0.2%	0.004605341

larger search space. However, the algorithm may take longer time to converge. The best replace worst strategy has a high selection pressure while the other strategies which contain a random element have a lower selection pressure. Therefore, even though the best replace worst strategy should perform well most of the time, the other strategies may also perform well for a large network with many alternative paths, where the search space is larger. In our experiment, this is evident in the result of 15 × 15 mesh networks where the random replace random strategy outperforms the best replace worst strategy.

5. Conclusion

This paper presents a study on the effect of migration strategies on the performance of a coarse-grained parallel GA-based shortest path routing algorithm. There are four migration strategies that have been evaluated which are best replace worst, best replace random, random replace worst and random replace random. Simulation result shows that best replace worst strategy does provide the best result most of the time. However, there are situations where the other strategies can perform just as well as, or even better than, the best replace worst strategy. It is also shown that the difference in performance between the best and the worst strategy is very low.

REFERENCES

[1] J. F. Kurose and K. W. Ross, "Computer Networking: A Top-Down Approach Featuring the Internet," 4th Edition, Addison-Wesley, Boston, 2007.

[2] M. Munetomo, N. Yamaguchi, K. Akama and Y. Sato, "Empirical Investigations on the Genetic Adaptive Routing Algorithm in the Internet," *Proceedings of the Congress on Evolutionary Computation*, Seoul, 27-30 May 2001, pp. 1236-1243.

[3] C. W. Ahn and R. S. Ramakrishna, "A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 6, 1999, pp. 1-7.

[4] W. Liu and L. Wang, "Solving the Shortest Path Routing Problem Using Noisy Hopfield Neural Networks," *International Conference on Communications and Mobile Computing*, Kunming, 6-8 January 2009, pp. 299-302. doi:10.1109/CMC.2009.366

[5] H. A. Mukhef, E. M. Farhan and M. R. Jassim, "Generalized Shortest Path Problem in Uncertain Environment Based on PSO," *Journal of Computer Science*, Vol. 4, No. 4, 2008, pp. 349-352.

[6] M. Yusoff, J. Ariffin and A. Mohamed, "A Discrete Particle Swarm Optimization with Random Selection Solution for the Shortest Path Problem," *International Conference of Soft Computing and Pattern Recognition*, Paris, 7-10 December 2010, pp. 133-138. doi:10.1109/SOCPAR.2010.5685867

[7] A. A. A. Zakzouk, H. M. Zaher and R. A. Z. El-Deen, "An Ant Colony Optimization Approach for Solving Shortest Path Routing Problem with Fuzzy Constraints," 7*th International Conference on Informatics and Systems*, Cairo, 28-30 March 2010, p. 1.

[8] S. Yussof and H. S. Ong, "A Robust GA-based QoS Routing Algorithm for Solving Multi-Constrained Path Routing Problem," *Journal of Computers*, Vol. 5, No. 9, 2010, pp. 1322-1334. doi:10.4304/jcp.5.9.1322-1334

[9] S. Yussof, R. A. Razali and H. S. Ong, "An Investigation of Using Parallel Genetic Algorithm for Shortest Path Routing Problem," *Journal of Computer Science*, Vol. 7, No. 2, 2011, pp. 206-215. doi:10.3844/jcssp.2011.206.215

[10] S.-C. Lin, E. D. Goodman and W. F. Punch, "Investigating Parallel Genetic Algorithms on Job Shop Scheduling Problems," *Lecture Notes in Computer Science*, Vol. 1213, 1997, pp. 383-393.

[11] Y. J. Cao, "Application of Parallel Genetic Algorithms to Economic Dispatch—Effects on Routing Strategies on Algorithms' Performances," *Automation of Electric Power System*, Vol. 26, No. 13, 2002, pp. 20-24.

[12] D. E. Goldberg, "Genetic Algorithm in Search, Optimization and Machine Learning," Addison-Wesley, Boston, 1989.

[13] E. C. Paz and D. E. Goldberg, "Efficient and Accurate Parallel Genetic Algorithms," Kluwer Academic Publications, Dordrecht, 2001.

[14] B. M. Waxman, "Routing for Multipoint Connections," *IEEE Journal on Selected Areas in Communications*, Vol. 6, No. 9, 1998, pp. 1617-1622. doi:10.1109/49.12889

