International Journal of the Physical Sciences Vol. 6(33), pp. 7524 - 7539, 9 December, 2011
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS11.613
ISSN 1992 - 1950 ©2011 Academic Journals

Full Length Research Paper

Performance analysis of genetic algorithm (GA)-based multi-constrained path routing algorithm

Salman Yussof

College of Information Technology, Universiti Tenaga Nasional, Selangor, Malaysia. E-mail: salman@uniten.edu.my.

Accepted 15 September, 2011

To support networked multimedia applications, it is important for the network to provide guaranteed quality-of-service (QoS). One way to provide such services is for the network to perform QoS routing, where the path taken must fulfill a given set of constraints. Multi-constrained path (MCP) problem refers to the problem of finding a path through a network subject to multiple additive constraints. It has been proven that this problem is Non-deterministic Polynomial time (NP)-complete and therefore no exact algorithm can be found. As such, various heuristics and approximation algorithms have been proposed to solve the MCP problem. This paper proposed a solution to the MCP problem using genetic algorithm (GA). The effectiveness of the proposed algorithm is evaluated through simulation. The performance of the algorithm is then compared with an exact algorithm called the depth first search and a common shortest path algorithm called the Dijkstra's algorithm. The result of the simulation shows that the performance of the proposed algorithm is almost comparable to an exact algorithm, while at the same time can execute much faster. The proposed algorithm has also been shown to have good network link utilization and is able to scale well with network size.

Key words: Genetic algorithm, multi-constrained path problem, quality-of-service (QoS), routing.

INTRODUCTION

With the emergence of networked multimedia applications, the traditional routing method is no longer adequate. This is because multimedia-oriented applications require a different set of requirements than the one required by data-oriented applications. According to Daneshmand et al. (1997), there are three primary performance parameters for multimedia applications: Delay, mean opinion score and differential delay. From networking point of view, these performance parameters can be translated to network parameters such as delay, jitter and bandwidth. Therefore, to ensure that the requirements for multimedia applications can be fulfilled, it is important for the network to be able to consider the corresponding quality-of-service (QoS) parameters when performing data transfer. One way of achieving this is by implementing QoS routing.

In general, routing consists of two tasks. The first task is to distribute the state information to the network and the second task is to find a feasible path by executing a routing algorithm that uses the state information as its input. This paper focuses on the second task and assumes that all the nodes have the correct state information. In traditional routing, the goal of the routing algorithm is to find the least-cost path from sender to receiver. QoS routing, on the other hand, has a more complicated goal. There are two main goals that need to be achieved by the QoS routing algorithm (Ghosh et al., 2001; Chen and Nahrstedt, 1998a). The first goal is to find a path that satisfies the QoS requirements. Such path is called a feasible path. Data transfer can only be performed when a feasible path is found. The second goal is to optimize the global network resource utilization. The second goal is necessary so that the network can accommodate as many QoS requests as possible.

There are various QoS routing algorithms that have been proposed by researchers. The algorithms are normally developed to address different problems based on the composition rule. There are three main composition rules: additive, multiplicative and concave. The definition of the composition rules as specified by Wang and Crowcroft (1996), and Chen and Nahrstedt (1998a) are given as follows:

Let $d(i,j)$ be a QoS metric for link (i, j). For any path $p = (i,$

j, k, …, l, m), metric *d* is additive if:

d(p) = d(i,j) + d(j,k) + … + d(l,m)

Metric *d* is multiplicative if:

d(p) = d(i,j) x *d(j,k)* x *…* x *d(l,m)*

Metric *d* is concave if:

d(p) = min[d(i,j), d(j,k), … , d(l,m)]

Constraints associated with concave QoS parameters can be easily dealt with by pruning all links that do not satisfy the constraints (Wang and Crowcroft, 1996). Several researchers such as Ghosh et al. (2001), Wang and Crowcroft (1996) and Yang et al. (2001) have developed algorithms to deal with a concave parameter or a combination of a concave and an additive parameter (that is, bandwidth-delay constraint). Constraints associated with multiplicative QoS parameters can be converted to additive parameters by using logarithm. However, the problem of finding a path subject to constraints of two or more additive QoS parameters is not very easy because it has been proven to be NP-complete (Wang and Crowcroft, 1996). This paper will mainly focus on this type of problem, which is also commonly known as the multi-constrained path (MCP) problem.

Definition 1

Multi-constrained path (MCP) problem

Consider a network represented by a directed graph $G = (N, E)$, where *N* is the set of nodes and *E* is the set of links. Each link $(i, j) \in E$ is associated with *K* additive QoS metrics $w_k(i, j)$, k = 1, 2, … , *K* where all metrics are non-negatives. The problem is to find a path *p* from a source node *s* to a destination node *d* such that $w_k(p) = \sum_{(i,j)\in p} w_k(i,j) \le C_k$ for k = 1, 2, … , *K*, where C_k is the constraint for the *k*th QoS metric.

Various heuristics and approximation algorithms have been proposed to solve the MCP problem. However, most of them are based on well-known shortest path routing algorithms such as Dijkstra's Algorithm or Bellman-Ford algorithm. The idea is to convert the multiple QoS metrics into a single metric which can be easily solved using a common shortest path routing algorithm. The earliest work on solving the MCP problem can be traced back to Jaffe (1984), where he solved a two-constraint problem by combining the two link metrics into a single mixed link metric. Chen and Nahrstedt (1998b) proposed an approximation to the MCP problem by scaling down all the QoS metrics except one. The problem is then simplified to the problem of finding the shortest path with respect to a single metric (the one that is not scaled down). Mieghem et al. (2001) proposed an algorithm called SAMCRA where the total cost with respect to each QoS metric is calculated and the total path cost is determined by the QoS metric that gives the largest total cost. Other works that are also based on techniques similar to the ones aforementioned are Khadivi et al. (2004) and Xue et al. (2008). There are also researchers who attempted to solve the MCP problem using other approaches such as flooding (Yen et al., 2008) and vector converting (Dai and Liu, 2009). MCP algorithms utilizing AI techniques such as fuzzy logic (Jing et al., 2008) and mobile agent (Wei and Yi, 2009) have also been explored by researchers.

This paper proposes a QoS routing algorithm based on genetic algorithm (GA) for solving the MCP problem. GA is chosen because it is a general-purpose search and optimization algorithm suitable for problems that have one or more of the following characteristics (Mitchell, 1996):

1. The search space is large;
2. The search space is known not to be perfectly smooth and unimodal;
3. The space is not well understood;
4. The fitness function is noisy;
5. The task does not require a global optimum to be found.

The QoS routing problem fits the characteristics given previously. It is a search problem, where the main goal is to find a path that can fulfill the QoS requirement. The search space can be large for large networks. The search space is also not very smooth due to the varying types of network links and the varying parameters on each link. And finally, the QoS routing problem does not require a global optimum to be found. It only needs to find one feasible path, and there can be more than one of such path in the network. Due to the similarities between the characteristics of the QoS routing problem and the type of problem that the GA can solve well, it is highly likely that genetic algorithm is suitable to be used for solving the QoS routing problem.

LITERATURE REVIEW

Introduction to genetic algorithm

Genetic algorithm (GA) is a search algorithm that is inspired by the theory of genetics and natural selection (Holland, 1975; 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 generation 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.

The general outline of GA as described by Goldberg (1989) and Dumetrescu et al. (2000) is as follows:

1. Initialize the initial population. The initial population is normally created randomly or based on some heuristics.
2. Evaluate the fitness of each chromosome on the population.
3. Perform selection. In this process, chromosomes are selected to be put into the mating pool. A selection scheme is utilized to choose the chromosomes.
4. Perform crossover. This process allows two chromosomes to exchange information and produce two new chromosomes. The parent chromosomes are selected randomly from the mating pool.
5. Perform mutation to the new chromosomes produced by crossover. This process randomly changes the content of a gene in a chromosome. This is done to create diversity in the population and allows the search of a new search space. Each gene will be considered for mutation with a certain probability.

The algorithm would then loop to step 2 and repeat all the subsequent steps until a termination condition is met. The termination condition can be one of the following:

1. A desired solution is found;
2. The population converges;
3. The maximum number of iteration has been reached.

GA-based routing algorithms

Many researchers have applied GA to various types of network routing problems. One of the earliest works on the use of GA for network routing is the work by Shimamoto et al. (1993). Shimamoto proposed a dynamic routing control based on GA to provide flexible real-time management of the dynamic traffic changes in broadband networks. The goal is to keep the traffic loss rate below a certain target value. He (1997) proposed a GA-based algorithm for the joint problem of selecting a route for each communicating node pair and a capacity value for each link the network. However, the proposed algorithm is more applicable to a network design problem rather than to a routing problem.

The first real GA-based routing algorithm was proposed by Munetomo et al. (1998). This algorithm addresses the problem of shortest path routing where delay is used as the link parameter. The proposed algorithm is not intended to replace standard shortest path algorithm such as Dijkstra's algorithm. Instead, the idea is to produce alternate paths which can be quickly used in the case of link failures. Ahn and Ramakrishna (2002) also proposed a GA-based routing algorithm for solving the shortest path routing problem. However, instead of trying to generate alternate paths, this algorithm is intended to compete directly with Dijktra's algorithm. They compared their results to that of Dijkstra's algorithm and found out that GA has two main advantages. The first one is that the GA algorithm is insensitive to variations in network topologies with respect to route optimality and convergence speed. The second one is that the proposed GA-based routing algorithm is scalable in the sense that the real computation time does not increase very much as the network size gets larger. Other researchers who also used GA to solve the shortest path routing problem are Sinclair (1998) and Hamdan and El-Hawary (2002).

With the rise of the importance of QoS routing, GA researchers have directed their attention to solving QoS routing problems. One of the earliest works on GA-based QoS routing algorithms comes from Xiang et al. (1999). Xiang proposes a GA-based QoS routing algorithm to solve a routing problem subject to four different QoS metrics: Delay, bandwidth, loss-rate and jitter. Wang and Wang (2001) and Riedl (2002) proposed GA-based QoS routing algorithms for solving the delay-bandwidth-constraint routing problem. On top of finding a feasible path, these two algorithms are also designed to optimize network resource utilization. Barolli et al. (2002) attempted to solve the problem of QoS routing subject to two QoS metrics which are delay and transmission success rate. Koyama et al. (2004) took Barolli's work further by introducing a multi-purpose optimization method that would further improve its performance.

Even though all the algorithms discussed previously use GA, they all vary in terms of the details of their implementation. In GA, each part of the algorithm such as the genetic encoding, the selection scheme, the crossover and mutation operations and the fitness function can be implemented in many different ways. In terms of genetic encoding, there are three types of encoding commonly used in network routing problem. Researchers such as Munetomo et al. (1998) and Ahn and Ramakrishna (2002) use a list of node IDs from source to destination to represent the chromosome. On the other hand, researchers such as Wang and Wang (2001) and Xiang et al. (1999) use a binary string to represent the GA chromosome. A matrix is used to represent all the possible links between any two nodes. A binary 1 is assigned to a value in the matrix if there is link between the two nodes, and 0 otherwise. Barolli et al. (2002) and Koyama et al. (2004) model the network as a tree and the GA chromosome is made up of a binary string that represents the junctions in the tree. Since different algorithms use different types of genetic encoding, the crossover and mutation operations vary

Table 1. Comparisons between GA-based routing algorithms.

Algorithm	Routing problem	Genetic encoding
Shimamoto et al. (1993)	Minimize call loss rate	A string of route numbers for all paths
He (1997)	Minimize delay and cost	A combination of line type and route serial number
Munetomo et al. (1998)	Shortest path	A list of node IDs in the path
Sinclair (1998)	Shortest path	(Not mentioned)
Ahn and Ramakrishna (2002)	Shortest path	A list of node IDs in the path
Hamdan and El-Hawary (2002)	Shortest path	A binary matrix where each cross point represents a link
Xiang et al. (1999)	Bandwidth, delay, loss rate and jitter constrained	A binary string where each bit in the string represents a link in the network
Wang and Wang (2001)	Bandwidth-delay constrained	A binary string where each bit in the string represents a link in the network
Rield (2002)	Shortest path and bandwidth-delay constrained	(Not mentioned)
Barolli et al. (2002)	Delay and transmission success rate constrained	A binary string where each bit in the string represents a junction in the tree structure used to represent the network
Koyama et al. (2004)	Multi-constrained path	A binary string where each bit in the string represents a junction in the tree structure used to represent the network

between algorithms since they are highly dependent on the genetic encoding used. The fitness functions also vary between algorithms because they are derived from the routing problem to be solved and the focus of the algorithm. Table 1 summarizes the comparison between GA-based routing algorithms reviewed here with respect to the routing problem solved and the genetic encoding used.

All the GA-based routing algorithms discussed earlier are designed for traditional fixed networks. However, in recent years, GA-based routing algorithms have also successfully been applied to other types of networks. Among them are mobile ad-hoc networks (Barolli et al., 2010), wireless sensor networks (Nallusamy et al., 2010), wireless mesh networks (Hua et al., 2010), delay / disruption tolerant networks (Silva and Guardieiro, 2010) and optical networks (Tode et al., 2010).

DESCRIPTION OF THE PROPOSED GA-BASED MCP ROUTING ALGORITHM

Algorithm overview

The proposed algorithm is designed to find a feasible path from source node to a destination node given a set of additive QoS requirement. The outline of the proposed algorithm is as follows:

1. Randomly initialize the initial population;
2. Evaluate the population using a fitness function;
3. Create the mating pool, which consists of all the chromosomes in the current population;
4. Apply crossover operator several times to create *m* new children for the new population (where *m* is the population size). The parents are selected using the selection operator. Each pair of parent chromosome can only mate once and the children produced are only accepted into the new population if they are not similar to the previously produced children. This is done to diversify the search and discourage convergence;
5. If all the possible pair of parents have mated and there are still not enough children to populate the new population, select members from the previous population using the selection operator to fill in the new population.
6. Apply mutation operator on the chromosomes in the mating pool. Each chromosome has a certain probability to be mutated. Mutation result must not be the same as any of the chromosome in the current population or else it would be ignored.
7. Replace the worst (the one with the lowest fitness value) chromosome produced by the genetic operators (crossover and mutation) with the best chromosome in the previous population.
8. Repeat step 2 until a feasible path is found or the maximum number of iteration is reached.

The overall procedure of the algorithm is depicted in the flowchart shown in Figure 1.

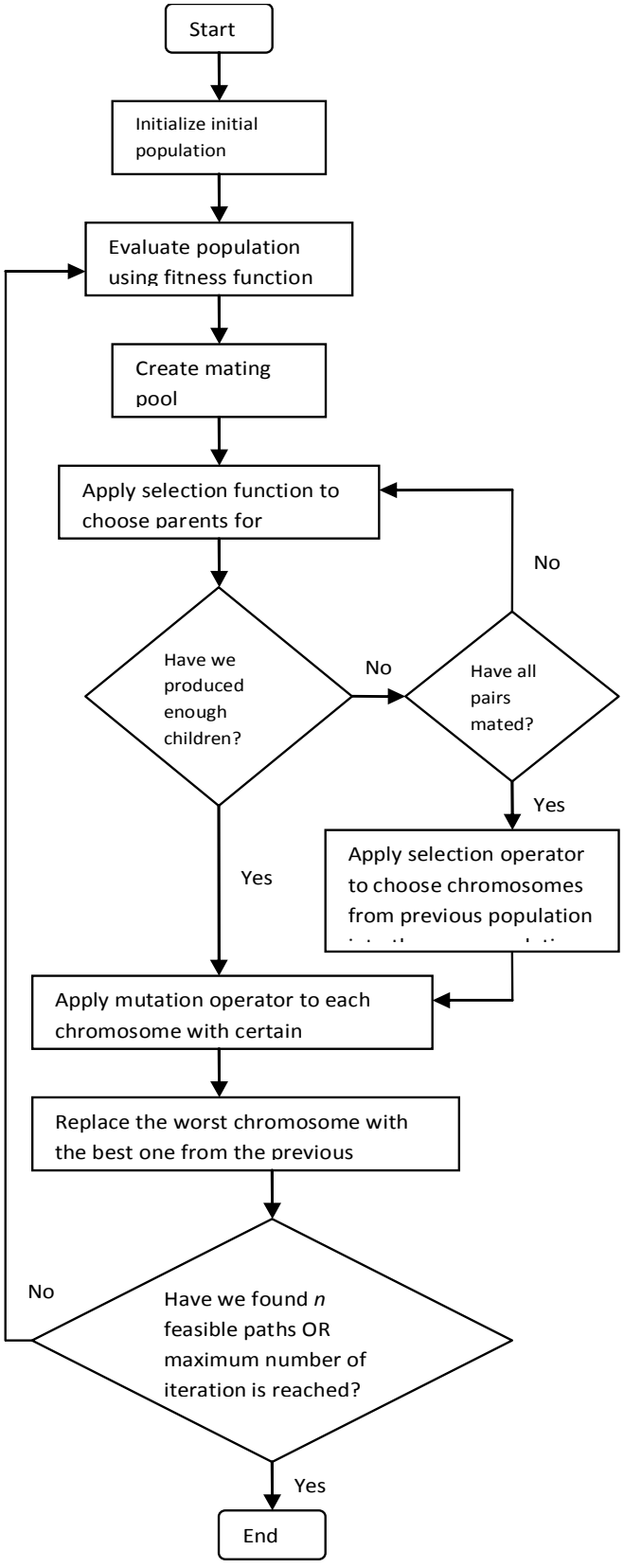

Figure 1. Overall procedure of the GA-based QoS routing algorithm.

As opposed to the algorithms proposed by Munetomo et al. (1998) and Ahn and Ramakrishna (2002) where the objective of the algorithm is to find the shortest path, the algorithm proposed here is designed to search a large search space for a feasible path. Therefore, this algorithm will not run until convergence. In fact, convergence is discouraged by not allowing similar chromosomes to be in the population. When a new chromosome is produced through a genetic operation (either crossover or mutation), the new chromosome is only included in the next generation if a similar chromosome does not yet exist. The proposed algorithm also includes a novel fitness function which is designed to solve the MCP routing problem.

Genetic representation

A communication network can be modeled as a directed graph $G(N,E)$, where N is the set of nodes representing the routers and E is the set of edges representing the links that connect between the routers. For a network supporting multiple QoS metrics, each edge (i,j) is associate with k independent metrics, $d_1(i,j)$, $d_2(i,j)$, $d_3(i,j)$, … , $d_k(i,j)$ where $d(i,j)$ is a real positive number.

The proposed algorithm is intended to be used in source routing where the sender executes the algorithm and finds a feasible path before the actual data can be sent. There are two methods by which the genetic encoding can be formulated. The first method is to follow the traditional GA where a chromosome is encoded as a string of binaries. The second method is to encode the chromosome using an encoding specific to the problem to be solved. For this algorithm, the latter method is chosen where each chromosome consists of a sequence of nodes that are in the path from sender to receiver. The first gene in the chromosome is always the sender and the last gene in the chromosome is always the receiver. Figure 2 shows an example of a small network that consists of six nodes where each link has two QoS metrics associated with it. An application requesting a connection between A and F can have its request fulfilled by two different paths: A→B→C→F and A→D→E→F. These two paths can be encoded as two different chromosomes:

Chromosome 1 (path 1): [A B C F]
Chromosome 2 (path 2): [A D E F]

The top path has a total QoS metric values of (8.11) and the bottom path has a total QoS metric values of (21.15). For a QoS aware application, any connection request made is accompanied by a set of QoS requirement. For example, if a multimedia application requested a connection with a QoS requirement of (15.13), it can be only fulfilled by the top path [A B C F].

Since different paths may have different number of intermediate nodes, the chromosomes will be of variable

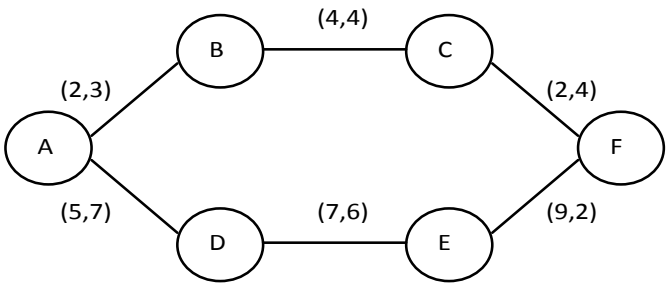

Figure 2. A small network with links that have two QoS metrics (k = 2).

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 population

In the beginning, the 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 depends on the population size.

The algorithm used to generate the random path is adapted from Ahn and Ramakrishna (2002). The algorithm goes as follows:

1. Start from the sending node;
2. Randomly choose, with equal probability, one of the nodes that are connected to the current node;
3. If the chosen node has not been visited before, mark that node as the next node in the path. Otherwise, find another node;
4. 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 receiving 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 (Dumetrescu et al., 2000). This information is then used to pick the chromosomes that will contribute to the formation of the next generation of solution. The fitness value is computed using a fitness function. The fitness function is highly dependant on the problem to be solved. For the MCP problem, the objective is to find a path that satisfies a set of QoS requirement as defined in Definition 1. In general, the smaller the total QoS values are on the links along a path from source to destination, the larger the possibility for the path to satisfy the given QoS requirement. Therefore, the fitness function must minimize the total cost C_T, which is defined as follows:

$$C_T = \sum_{k=0}^{K-1} \sum_{l=0}^{L-1} c_{kl} \quad (1)$$

where c_{kl} represents the kth QoS metric on link l, K represents the number of QoS metric on each link and L represents the total number of links in the path. However, since the range of values of each QoS metric can vary depending on the type of the metric, simply adding up all the QoS metrics would cause the result to be dominated by QoS metrics that have larger values compared to the others. Therefore, before the QoS metrics can be added up, each QoS metric must be normalized as follows:

$$c_{kl}' = \frac{c_{kl}}{\sum_{r=0}^{R-1} \sum_{j=0}^{L-1} c_{kj}} \quad (2)$$

where R represents the total number of possible paths from source to destination. The term $\sum_{r=0}^{R-1} \sum_{k=0}^{L-1} c_{kj}$ adds up all the kth QoS metric on each link along each possible path.

Based on the argument aforementioned, given a population of size p, the fitness function is defined as follows:

$$f_i = \frac{\sum_{j=0}^{K-1} \left[\frac{c_{ij}}{c_j} \right]}{K} \quad (3)$$

where f_i represents the fitness value of the chromosome i, K represents the number of QoS metric on each network link, $c_{ij} = \sum_{l \in i} c_{ijl}$ represents the sum of the jth QoS metric for all links l within chromosome i, and $c_j = \sum_{i=0}^{p-1} c_{ij}$ is obtained by adding up the jth QoS metrics for all links of all the chromosomes in the population. The division with K is done to normalize the whole result so that the resulting fitness value is between 0 and 1, where a lower value indicates a better path. The total fitness values

from all the chromosomes in the population would add up to 1.

As an example, assume the network depicted in Figure 2. There are two paths that can be used to travel from node A to node F. Therefore, we will assume that the population has two chromosomes representing these two paths. The fitness values for the top path (f_0) and the bottom path (f_1) are computed as follows:

$$f_0 = \frac{\frac{8}{29} + \frac{11}{26}}{2} = 0.35$$

$$f_1 = \frac{\frac{21}{29} + \frac{15}{26}}{2} = 0.65$$

In the example aforementioned, the value given by the fitness function indicates that the top path is the better path. This can be verified to be correct considering that the top path generally has lower QoS metric values. It is important to point out that the generality of the proposed algorithm which enables it to be applied to any *k*-constrained MCP QoS routing problem is due to the generality of this fitness function.

Selection

In order to generate the next generation of solutions, a mating pool that consists of the members of the current population is created. The chromosomes in the mating pool are then subjected to genetic operations such as crossover and mutation. Parents for the crossover operation are selected using a selection operator. The idea behind the selection operator is to allow chromosomes with better fitness values a higher chance to reproduce through crossover operation. Parents with good fitness values are expected to produce children with even better fitness values.

According to Dumetrescu et al. (2000), there are several categories of selection scheme. Among them are proportional selection, rank-based selection and tournament selection. For each category, there are several variations of the selection schemes. This algorithm uses a type of proportional selection called stochastic sampling with replacement, implemented using roulette wheel (Monte Carlo) method. To select two parents, this operation is performed twice. A pair of parent chromosomes can only be selected once.

Crossover

The first genetic operation done to the chromosomes in the mating pool is crossover. The idea behind crossover is to create an information exchange between two chromosomes (Dumetrescu, 2000). By doing so, the algorithm will explore new paths and hopefully be able to find better paths in the process.

In order to perform crossover, two chromosomes from the mating pool will be selected using the selection operator. These two chromosomes will become the parent chromosomes. In an iteration, a pair of parents can only be selected for crossover once. 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 sending and receiving 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. Since the number of possible crossover point in a pair of chromosome is normally very small, crossover is always performed whenever there is at least one crossover point (that is, crossover probability, $p_c = 1$). The actual crossover operation is similar to the one used by Munetomo (1998) and Ahn and Ramakrishna (2002). 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 sending node and receiving 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.

It is possible that loops may occur after crossover operation is performed. Loops in a chromosome can be repaired by performing a search along the chromosome to find repeated nodes (Ahn and Ramakrishna, 2002). The nodes in between the repeated nodes are then eliminated. For example, assume that we have the following chromosome that contains a loop:

Chromosome with loop: [A B C G H I K G U W Z]

In this case, there are two G nodes in the chromosome which signifies that the path contains a loop. This chromosome can be fixed by eliminating one of the G node and all the other nodes in between the two G nodes. The loop-free chromosome would become like this:

Loop-free chromosome: [A B C G U W Z]

The resulting chromosome can be searched again just in case there are multiple loops in the chromosome.

Mutation

The objective of mutation operation is to create diversity in the population (Dumetrescu, 2000). Mutation would allow the algorithm to search on the areas of the search space that was previously unknown. Each chromosome in the population would have a certain probability to be mutated. This probability is referred to as the mutation rate. For optimal result, the value for the mutation rate has to be set correctly.

The actual mutation operation is adapted from Ahn and Ramakrishna (2002). 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 (that is, 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 … 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.

EXPERIMENTS

To evaluate the performance of the algorithm, a number of simulations had been performed using OMNeT++. The simulations were run on a workstation with 3.0 GHz Xeon processor and 1GB of RAM. The performance of the GA-based QoS routing algorithm will be compared with two other algorithms. The first algorithm is an exact algorithm implemented using depth first search (DFS). The DFS algorithm will search every single path in the network to find a feasible path and is guaranteed to find a feasible path if such path exists. In this DFS implementation, the algorithm terminates as soon as a feasible path is found. Even though the DFS algorithm is confirmed to find a feasible path if such paths exist, it has an asymptotic exponential worst-case time complexity and therefore is not suitable to be used in practice. The second algorithm to be used for comparison is an algorithm that uses a single mixed metric. Even though a single mixed metric does not contain enough information to determine whether QoS requirements can be satisfied, it can reduce the time complexity because a single source single destination shortest path algorithm such as Dijkstra's algorithm can be employed (Khadivi et al., 2004). For the experiments, the mixed metric for a link, *w(e),* is simply defined as:

$$w(e) = \frac{1}{n}\sum_{i=0}^{n} w_i \quad (4)$$

where n is the number QoS parameters that the link has. With the use of a single link parameter, a path is selected by using Dijkstra's algorithm. The selected path is then evaluated to check whether it fulfills the QoS requirements.

There are two network topologies used for the simulations. The first topology is a 10×10 mesh network and the second topology is a network referred to as the NTT backbone (a network topology included in the OMNeT++ software package) as shown in Figure 3. During the simulation, nodes will be selected randomly to make a connection request to another node (also selected randomly) with a particular QoS constraint. The routing algorithms are then executed to search for a path to that destination that can fulfill the given constraint.

Four different types of simulation had been performed. The first simulation is performed to find the proper population size to be used in all the other simulations. The second simulation is to evaluate the performance of the proposed algorithm in terms of success ratio. The third simulation is to evaluate network resource utilization. The fourth simulation is to see how the algorithm scales with network size in terms of processing speed. For all the simulations, the simulation time used for each run is 3000 s.

RESULTS

Finding the proper population size

This simulation was done to find out the proper population size to be used in all the other simulations. Population size is one of the important parameters in GA. A larger population size would cause the algorithm to take less number of iterations to find a feasible path and at the same time it may be able to find better results. However, larger population size will also take more memory and computation time. A good population size should give a balance between the number of iterations taken, the quality of the results and the system resources used.

In this simulation, only the NTT backbone network was used. The simulation was run with population size ranging from 5 to 50. For each run, the total number of iterations taken by each node for the whole duration of simulation, the success ratio and the total time taken by the routing algorithm to be executed by all nodes are recorded. Success ratio is defined as follows:

$$\text{Success ratio} = \frac{\text{Number of requests successfully routed}}{\text{Total number of routing requests}}$$

The result of this simulation is given in Figures 4 and 5. Figure 4 shows the total number of iterations taken from all the nodes for the duration of the simulation. As expected, the number of iterations required gets smaller as the population size grows larger. The number of iterations decreases logarithmically. Figure 5 shows the success ratio for different population size. The success ratio increases logarithmically as the population size grows larger. This shows that as the population size increases, the quality of the solution gets better. From these two results, it can be concluded that the number of iterations required and the quality of the solution do get

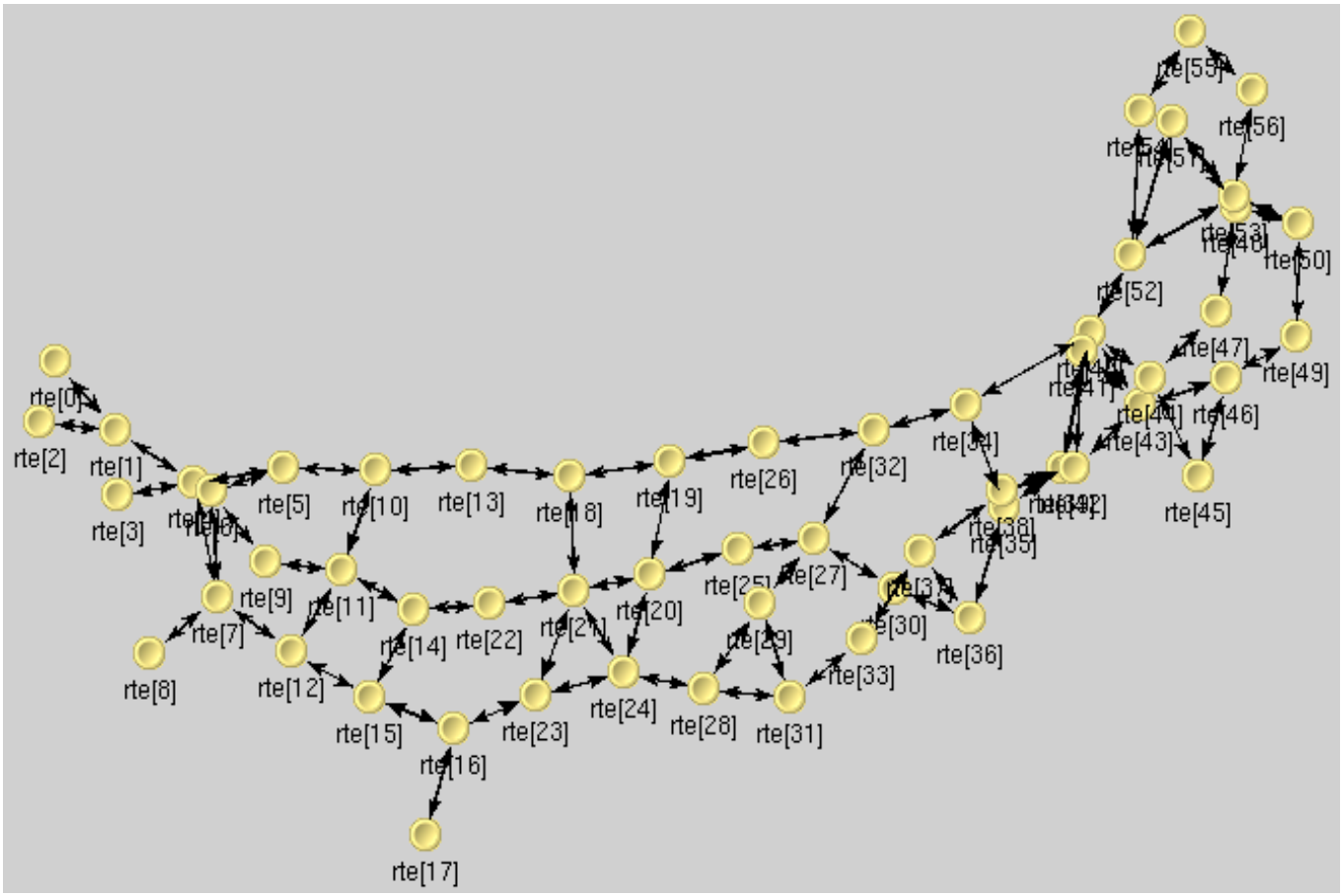

Figure 3. NTT backbone network.

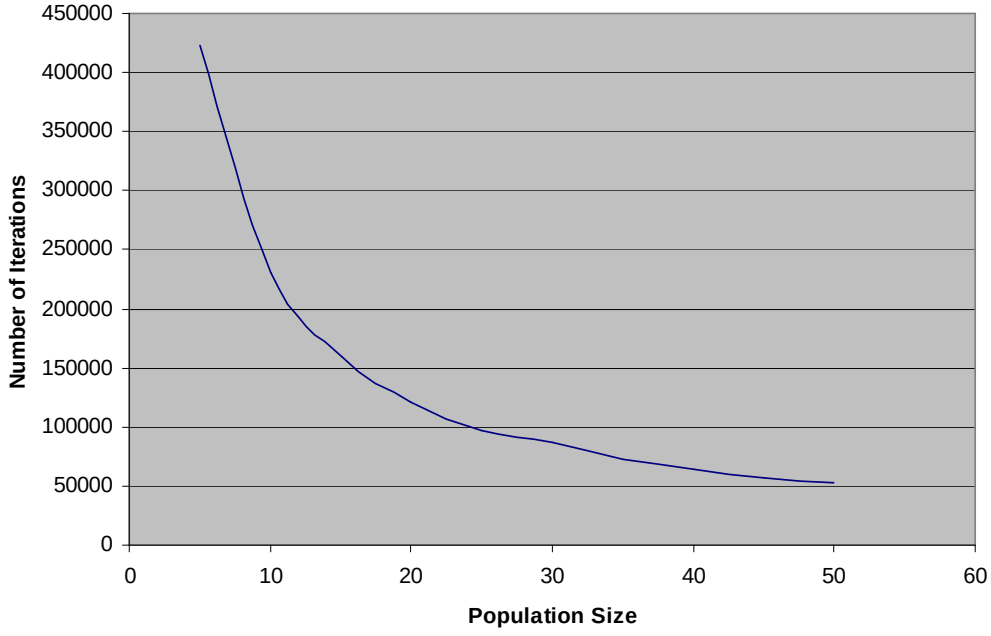

Figure 4. Total number of iterations for the duration of simulation for different population size.

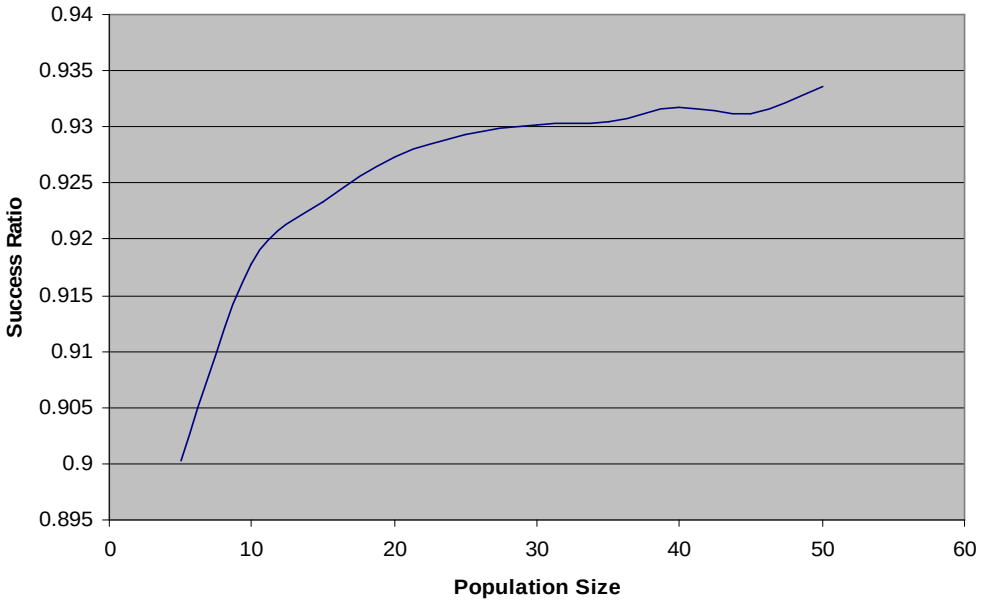

Figure 5. Success ratio for different population size.

Table 2. Ranges of link parameters and the correlation between them.

Positive correlation	No correlation	Negative correlation
$w_1(i,j)$ ~ [1,50]		$w_1(i,j)$ ~ [1,50]
$w_2(i,j)$ ~ [1,100]	$w_1(i,j)$ ~ [1,100]	$w_2(i,j)$ ~ [100,200]
OR	$w_2(i,j)$ ~ [1,200]	OR
$w_1(i,j)$ ~ [50,100]		$w_1(i,j)$ ~ [50,100]
$w_2(i,j)$ ~[100,200]		$w_2(i,j)$ ~ [1,100]

better with population size. But, as the population size grows larger, the percentage of improvement gets smaller.

Based on the result of this simulation, we have decided to use population size 25 for the rest the paper. This is because for population size larger than 25, the percentage of improvement in success ratio and the number of iterations taken is very low.

Simulation results for evaluating success ratios

In this simulation, both the 10×10 mesh network and the NTT backbone network were used. For each link in the network, two additive QoS parameters w_1 and w_2 are randomly generated. These parameters are selected from uniform distribution under several types of correlation between them. The range of the parameters and the correlation between them can be found in Table 2. The source and destination of a request are randomly generated. The QoS constraints C_1 and C_2 for each routing requests are selected from uniform distribution. The population size for GA is set to 25 and the algorithm iterates to a maximum of 100 iterations before it reports failure.

For each topology, three simulations were performed for each of the correlation type. The three simulations differ in terms of the range of the QoS constraints given for routing requests: low constraints, medium constraints and high constraints. For low constraints, not many routing requests can be fulfilled, leading to low success ratio. The reverse is true for high constraints. The range of the QoS parameters for the three constraints can be found in Table 3. For each simulation, the results are obtained from three different runs, where for each run the network is given new link parameters generated using random seeds. The total number of requests from the three runs for the NTT backbone network is around 50,000 and for the mesh network, the total number of requests is around 89,000.

Figures 6 and 7 shows the results of the simulation for the two networks and for each of the three types of

Table 3. Ranges of constraints for QoS requests.

Constraint	Constraint for QoS Parameter 1	Constraint for QoS Parameter 2
Low	C_1 ~ [100,300]	C_2 ~ [200,500]
Medium	C_1 ~ [300,500]	C_2 ~ [600,1000]
High	C_1 ~ [500,700]	C_2 ~ [1000,1500]

correlation. As expected, the success rate depends a lot on the constraints for the QoS requests. Regardless of the routing algorithm used and the correlation between link parameters, a higher constraint will definitely give a higher success ratio.

Among the three algorithms, the success ratio of DFS is the highest since it is a brute force algorithm that search all possible paths that guarantees to find a feasible path if it exists. GA however performs very close to the performance of DFS. Tables 4 and 5 show the percentage of GA's success ratio as compared to DFS for the NTT backbone network and the mesh network, respectively. For the NTT backbone network, the percentage is at least 99.06%, which shows that GA is highly successful in its routing decision. For the mesh network, however, the percentage can be a bit lower. The lowest is around 91.58%. The Dijkstra's algorithm performed the worst due to the use of a single link parameter that does not contain enough information to guide the search of a feasible path during the routing process.

The correlation between the link parameters also seems to have some effect on the success ratio of all the three algorithms regardless of the type of constraints, although not very much. Link parameters with positive correlation seem to give the highest success ratio, followed by no correlation and negative correlation.

Simulation results for evaluating resource utilization

In this simulation, the three algorithms are run on both the NTT backbone network and the mesh network. For all the runs, the link parameters assigned have no correlation with each other and the QoS requests are of high constraints. The objective of this simulation is to monitor resource utilization. Optimizing resource utilization is one of the goals in QoS routing. Several other MCP algorithms such as Baoxian and Mouftah (2003) assign one of the QoS parameters to represent resource utilization. The algorithm is then designed to optimize this one parameter. However, assigning a single parameter to represent resource utilization may not be feasible in practice. Therefore, in this simulation, resource utilization is measured by the distribution of network links usage. The number of times each link is used throughout the simulation is recorded. Link utilization is measured by counting the number of times each link is used and computing the standard deviation of link usage. The standard deviation is calculated as follows:

$$\sigma = \sqrt{\frac{1}{L}\sum_{l=1}^{L}(u_l - \overline{u})^2} \quad (5)$$

where L represents to total number of links in the network, u_l represents the number of times link l is used and $\overline{u}$ is the mean of link utilization which is calculated as follows:

$$\overline{u} = \frac{1}{L}\sum_{i=1}^{L} u_l \quad (6)$$

It is assumed that the mean, $\overline{u}$, is the optimum solution where all the links in the network are equally utilized. A low standard deviation would indicate a distributed link usage closer to the mean and this, in turn, indicates a better network link utilization.

Figure 8 shows the result of this simulation. Out of the three algorithms, DFS gives the highest standard deviation, followed by GA and Dijkstra's algorithm. The standard deviation of GA and Dijkstra's algorithm, however, is very close. The high standard deviation of DFS shows that the link usage is very much concentrated on certain links only and it does not use the network links evenly. GA, on the other hand, with its lower standard deviation shows that it can make more efficient use of the network links. Even though the standard deviation of Dijkstra's algorithm is lower, its low success ratio as discovered in the previous experiment does not make it a good MCP routing algorithm.

Simulation results for evaluating processing speed for various network size

In this simulation, the three algorithms are run on mesh networks with different sizes: 10×10 (100 nodes), 15×15 (225 nodes) and 20×20 (400 nodes). For all the runs, the link parameters assigned have no correlation with each other and the QoS requests are of high constraints. At the end of the simulation, the total amount of processing time of all the nodes is recorded. The objective here is to see how the well the algorithm scales with network size in terms of processing speed.

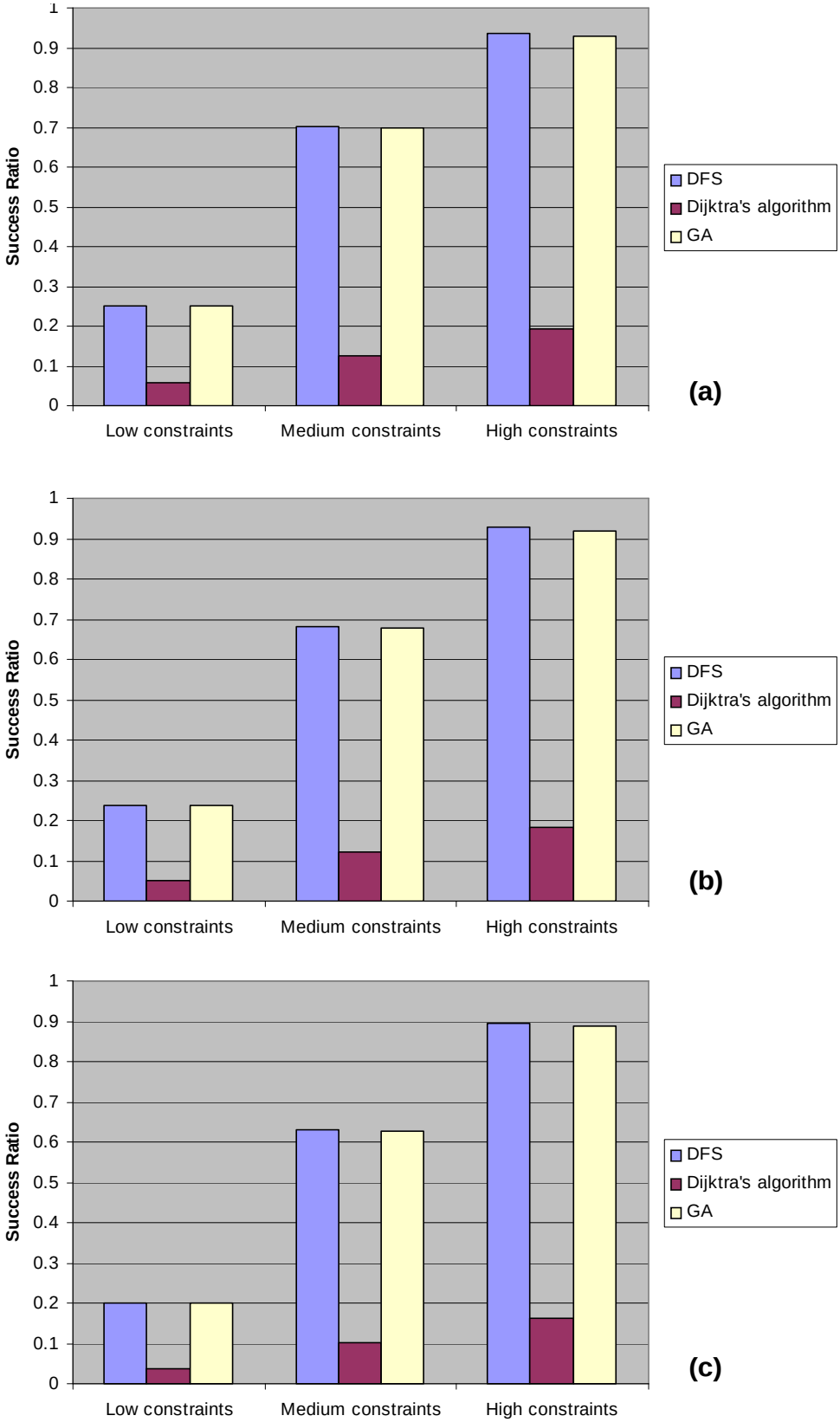

Figure 6. Success ratio for the three algorithms on NTT backbone network with respect to different link correlations a); no correlation (b), and negative correlation (c)] and QoS request constraints.

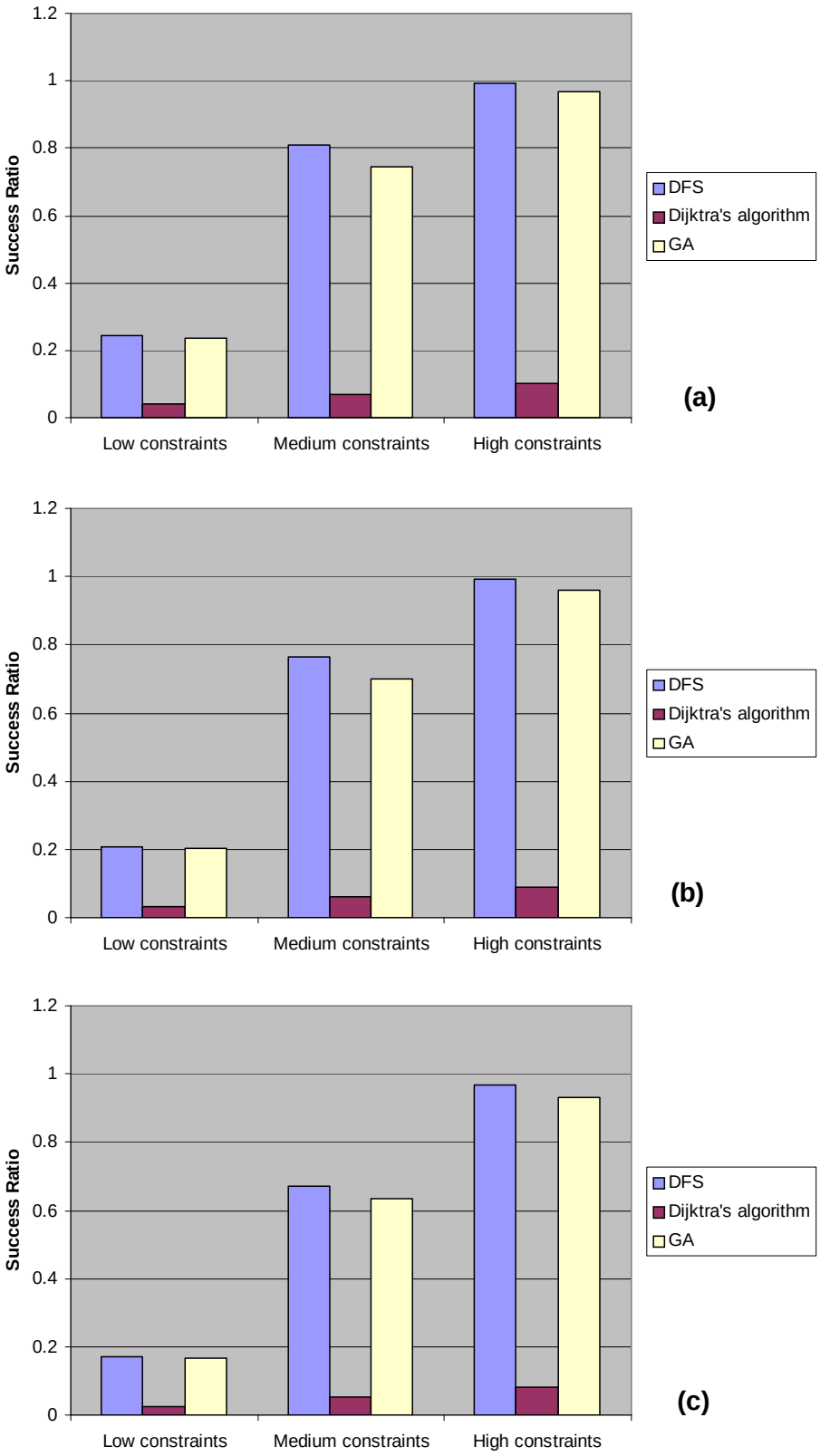

Figure 7. Success ratio for the three algorithms on 10×10 mesth network with respect to different link correlations [positive correlation [(a); no correlation (b), and negative correlation (c)] and QoS request constraints.

Table 4. Percentage of GA success ratio as compared to DFS for NTT backbone.

Correlation	Low constraint (%)	Medium constraint (%)	High constraint (%)
No correlation	99.83	99.23	99.06
Negative correlation	99.91	99.53	99.21
Positive correlation	99.81	99.30	99.18

Table 5. Percentage of GA success ratio as compared to DFS for 10×10 mesh network.

Correlation	Low constraint (%)	Medium constraint (%)	High constraint (%)
No correlation	97.59	91.58	96.78
Negative correlation	99.11	94.46	96.34
Positive correlation	96.48	91.73	97.41

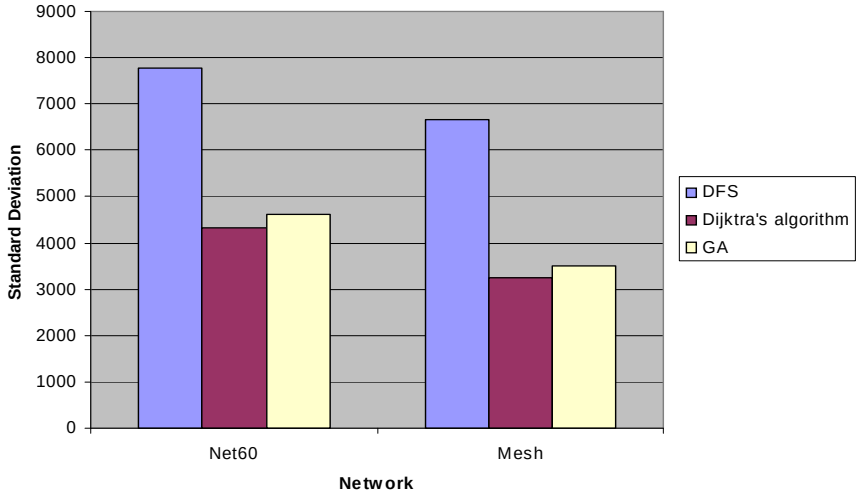

Figure 8. Link utilization for the three algorithms.

The result of this simulation is shown in Figure 9. The processing time for all three algorithms increases with larger network size. However, the GA-based routing algorithm performs much faster compared to DFS for larger networks. Even though the speed is not as fast as Dijkstra's algorithm, the slower speed is justified due to the much better performance that it has.

DISCUSSION

Evaluating the performance of an MCP routing algorithm is difficult because the performance can be affected by various factors such as the range of values for QoS constraints, the range of values for the QoS parameters of the network links, the correlation between the QoS parameters on the network links, the topology of the network and also the network size. This paper tries to take a number of these factors into account during simulation to ensure that the algorithm works for various different cases.

The use of GA also provides its own challenges. As mentioned previously, there are many parts of GA that can be implemented differently such as the genetic encoding, the genetic operators and the details of the genetic operations. And then, there are various parameters such as the population size, mutation rate, crossover rate and maximum iteration that can affect the performance of the algorithm. All of these can affect the performance of the algorithm. In this paper, only the population size is determined experimentally. A more detailed study on the effect of the other GA parameters is presented in Yussof and Ong (2008).

The results of the experiments show that the proposed GA-based MCP routing algorithm is able to strike a balance between the slow, but highly accurate DFS

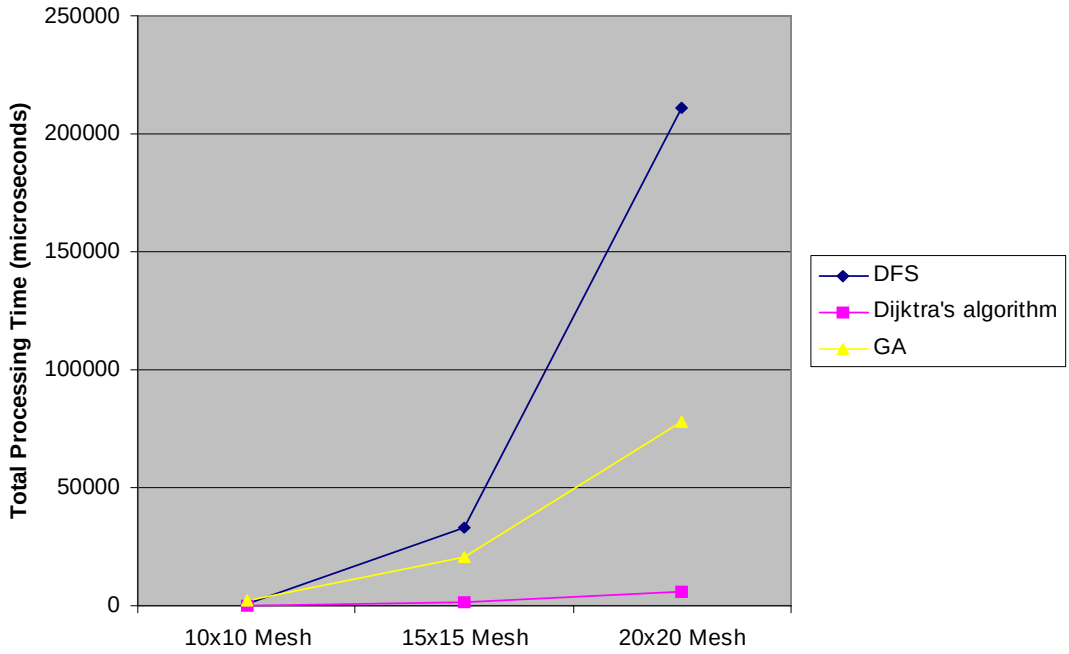

Figure 9. Processing speed for various network sizes.

algorithm and the fast, but inaccurate Dijkstra's algorithm. The results also show that the proposed algorithm can provide high accuracy, with relatively faster execution time compared to the DFS algorithm. In addition to that, the proposed algorithm has been shown to provide relatively good network resource utilization. These two characteristics would allow the proposed algorithm to fulfill the two objectives of QoS routing and show that using GA is a promising approach for solving the MCP routing problem.

Conclusion

This paper proposed a GA-based QoS routing algorithm for solving the MCP problem. The chromosome in this algorithm consists of a series of nodes that is in the path from sender to receiver. Based on the simulation, this algorithm has been shown to perform well to achieve the two objectives of QoS routing which are to find a feasible path and to optimize network resource utilization. The success ratio achieved is very close to that of an exact algorithm, where in all the simulations, the success ratio is more than 99% as compared to the exact algorithm regardless of the correlation between the link parameters and the type of constraints generated by the QoS requests. In addition to having a high success ratio, this algorithm has been shown to provide a better link utilization and faster processing time as compared to the exact algorithm. The result is consistent in the two network topologies used for the simulations.

REFERENCES

Ahn CW, Ramakrishna RS (2002). A genetic algorithm for shortest path routing problem and the sizing of populations. IEEE Transactions on Evolutionary Computing, 6: 566-579.

Baoxian Z, Mouftah HT (2003). A stateless QoS routing algorithm subject to multiple constraints. In proceedings of IEEE International Conference on Communications, 3: 1870-1874.

Barolli A, Takizawa M, Xhafa F, Barolli L (2010). Application of genetic algorithms for QoS routing in mobile ad hoc networks: a survey. In proceedings of International Conference on Broadband, Wireless Computing, Communication and Applications, pp. 250-259.

Barolli L, Koyama A, Matsumoto K, Suganuma T, Shiratori N (2002). A genetic algorithm based routing method using two QoS parameters. In proceedings of 13th International Workshop on Database and Expert Systems Applications, pp 3-7.

Chen S, Nahrstedt K (1998a). An overview of quality of service routing for next-generation high-speed networks. IEEE Network. 12(6): 64-79.

Chen S, Nahrstedt K (1998b). On finding multi-constrained path. In proceedings of IEEE International Conference on Communication, 2: 874-879.

Dai FS, Liu AJ (2009). A multi-constrained quality-of-service routing algorithm based on vector converting. In proceedings of 5th International Conference on Wireless Communications, Networking and Mobile Computing, pp.1-4.

Daneshmand MF, Roy RR, Savolaine CG (1997). Framework and requirements of quality of service for multimedia application. Intelligent Information System, pp 466-474.

Dumetrescu D, Lazzerini B, Jain LC, Dumitrescu A (2000). Evolutionary Computing. The CRC Press.

Ghosh D, Sarangan V, Acharya R (2001). Quality of service routing in IP networks. IEEE Transactions on Multimedia. 3(2): 200-208.

Goldberg GE (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley.

Hamdan M, El-Hawary ME (2002). Hopfield-Genetic approach for solving the routing problem in computer networks. In proceedings of Canadian Conference on Electrical and Computer Engineering, 2: 823-827.

He C (1997). Route selection and capacity assignment in computer communication networks based on genetic algorithm. In proceedings of IEEE International Conference on Intelligent Processing Systems. 1: 548-552.

Holland JH (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press.

Hua J, Liping Z, Yanxiu L, Min Z (2010). Multi-constratined QOS routing optimization of wireless mesh network based on hybrid genetic algorithm. In proceedings of International Conference on Intelligent Computing and Integrated Systems, pp. 862-865.

Jaffe JM (1984). Algorithms for finding paths with multiple constraints. Networks. 14: 95-116.

Jing Z, Xuefen C, Guan L, Hongxia L (2008). Service-aware multi-constrained routing protocol with QoS guarantee based on fuzzy logic. In proceedings of 22nd International Conference on Advanced Networking and Applications Workshop, pp. 762-767.

Khadivi P, Samavi S, Todd TD, Saidi H (2004). Multi-constraint QoS routing using a new single mixed metric. In proceedings of IEEE International Conference on Communications. 4: 2042-2046.

Koyama A, Barolli L, Matsumoto K, Apduhan BO (2004). A GA-based multi-purpose optimization algorithm for QoS routing. In proceedings of 18th International Conference on Advanced Information Networking and Applications, 1: 23-28.

Mieghem PV, Neve HD, Kuipers F (2001). Hop-by-hop quality of service routing. Computer Networks, 37: 407-423.

Mitchell M (1996). An Introduction to Genetic Algorithms. MIT Press.

Munetomo M, Yamaguchi N, Akama K, Sato Y (1998). A migration scheme for the genetic adaptive routing algorithm. In proceedings of IEEE International Conference on Systems, Man and Cybernatics. 3: 2774-2779.

Nallusamy R, Duraiswamy K, Muthukumar DA, Sathiyakumar C (2010). Energy efficient dynamic shortest path routing in wireless Ad hoc sensor networks using genetic algorithm. In proceedings of International Conference on Wireless Communication and Sensor Computing, pp 1-5.

Rield A (2002). A hybrid genetic algorithm for routing optimization in IP networks utilizing bandwidth and delay metrics. In proceedings of IEEE Workshop on IP Operations and Management, pp. 166-170.

Shimamoto N, Hiramatsu A, Yamasaki K (1993). A dynamic routing control based on a genetic algorithm. In proceedings of IEEE Conference on Neural Networks, pp 1123-1128.

Silva ER, Guardieiro P (2010). An efficient genetic algorithm for anycast routing in delay/disruption tolerant networks. IEEE Communication Letters. 14(4): 315-317.

Sinclair MC (1998). Minimum cost routing and wavelength allocation using genetic algorithm / heuristic hybrid approach. In proceedings of 6th IEE Conference on Telecommunications, pp 62-71.

Tode H, Hamada K, Murakami K (2010). ORGAN: Online route and Wavelength design based on Genetic Algorithm for OPS networks. In proceedings of Conference on Optical Network Design and Modeling, pp 1-6.

Wang X, Wang G (2001). An algorithm for QoS routing to optimize network resource utilization. In proceedings of International Conference on Info-tech and Info-net, 2: 474-479.

Wang Z, Crowcroft J (1996). Quality-of-service routing for supporting multimedia applications. IEEE J. Selected Areas in Comm., 14(7): 1228-1234.

Wei C, Yi Z (2009). A multi-constrained routing algorithm based on mobile agent for MANET networks. In proc. Int. Joint Conf. Art. Intell., pp. 16-19.

Xiang F, Junzhou L, Jieyi W, Guanqun G (1999). QoS routing based on genetic algorithm. Computer Communications. 22(15-16), pp 1392-1399.

Xue G, Zhang W, Tang J, Thulasiraman K. Polynomial time approximation algorithms for multi-constrained QoS routing. IEEE/ACM Transactions on Networking. 16(3): 656-669.

Yen YS, Chang RS, Chao HC (2008). Flooding- limited for multi-constrained quality-of-service routing protocol in mobile ad hoc networks. IET Communications, 2(7): 971-981.

Yussof S, Ong HS (2008). The Effect of GA Parameters on the Performance of GA-based QoS Routing Algorithm. In proceedings of 3rd International Symposium on Information Technology, pp 1-7.