A Robust GA-based QoS Routing Algorithm for Solving Multi-constrained Path Problem

Salman Yussof
College of Information Technology, Universiti Tenaga Nasional, Kajang, Selangor, Malaysia
Email: salman@uniten.edu.my

Ong Hang See
College of Engineering, Universiti Tenaga Nasional, Kajang, Selangor, Malaysia
Email: ong@uniten.edu.my

***Abstract*— To support networked multimedia applications, it is important for a 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 certain 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 NP-complete and finding an exact solution can be difficult. As such, various heuristics and approximation algorithms have been proposed to solve the MCP problem. However, the actual link metrics in a QoS-aware network is dynamic and may continuously change over time and since the path given by the routing algorithm is computed using the state information available to the router, which may or may not be up-to-date, it is possible that a feasible path returned by the algorithm may turn out to be no longer valid. This paper presents a GA-based QoS routing algorithm for solving the general *k*-constrained problem which has the capability to return multiple feasible paths in a single run. This makes the algorithm more robust in the case that the rate of change of state information in the network is higher than the rate of state information received by the router. Simulation results show that this algorithm consistently achieve higher feasibility ratio relative to existing well-known MCP routing algorithms when state information in the router lags behind the network.**

***Index Terms*— Genetic algorithm, multi-constrained path, QoS routing, robust routing.**

I. 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. [1], 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. Any data transfer performed without fulfilling the QoS requirement would be useless because the receiver may no longer be able to use the received data. 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 network parameters when performing data transfer. One way of achieving this is by implementing QoS routing.

In traditional routing, the main objective of the routing algorithm is to find the least-cost path from sender to receiver. QoS routing, on the other hand, has two objectives [2], [3]. The first objective 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 objective is to optimize the global network resource utilization. This is necessary so that the network can accommodate as many QoS requests as possible.

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. The first task is done by routing protocols such as RIP and OSPF, and in the case of QoS routing, a QoS routing protocol such as QoSPF is required [4]. In a data network such as the Internet, the link metric does not change very often. However, in a QoS-aware network which commonly runs networked multimedia applications, the link metrics such as delay and delay jitter can continuously change over time depending on the type and amount of multimedia data being transmitted. This makes it difficult for a router to have an up-to-date information of all the link metrics in the network [5]. As such, it is possible for a feasible path returned by the QoS routing algorithm to turn out to be not feasible when the actual data transfer is performed.

There are various QoS routing algorithms that have been proposed by researchers. The algorithms are

Manuscript received January 13, 2010; revised March 15, 2010; accepted March 16, 2010.

This work was supported by Universiti Tenaga Nasional.

Salman Yussof is with the Department of Systems and Networking, College of Information Technology, Universiti Tenaga Nasional, Malaysia (email: salman@uniten.edu.my).

Ong Hang See is with the Department of Electronics and Communication Engineering, College of Engineering, Universiti Tenaga Nasional, Malaysia (email: ong@uniten.edu.my).

doi:10.4304/jcp.5.9.1322-1334

normally developed to tackle 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 et al. [6] and Chen et al. [7] are given below:

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:

$$d(p) = d(i,j) + d(j,k) + \ldots + d(l,m)$$

Metric *d* is multiplicative if:

$$d(p) = d(i,j) \times d(j,k) \times \ldots \times d(l,m)$$

Metric *d* is concave if:

$$d(p) = min[d(i,j), d(j,k), \ldots, d(l,m)]$$

Constraints associated with concave QoS metrics can be easily dealt with by pruning all links that do not satisfy the constraints [6]. Several researchers such as the ones in [2], [6] and [8] have developed algorithms to deal with a concave metric or a combination of a concave and an additive metric (i.e. bandwidth-delay constraint). Constraints associated with multiplicative QoS metrics can be converted to additive metrics by using logarithm. However, the problem of finding a path subject to constraints of two or more additive QoS metrics is not very easy and it has been proven to be NP-complete [6]. 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 $w_k(i, j)$, $k = 1, 2, \ldots, K$ 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 $k = 1, 2, \ldots, K$, where C_k is the constraint for the *k*th QoS metric.

In this paper, we propose a general *k*-constrained QoS routing algorithm for solving the MCP problem. In addition to solving the MCP problem, the proposed algorithm is also able to produce multiple feasible paths and this enables the algorithm to become more robust in the case that the actual state information in the network has changed since the last state information update received by the router. If the first feasible path found turns out to be no longer feasible, there are other alternative paths that can be tried without having to execute a new routing computation. This will increase the success of finding a feasible path that can fulfill a particular QoS requirement.

The rest of the paper is organized as follows. In Section II, a literature review on genetic algorithm and MCP routing algorithms are presented. The proposed GA-based QoS routing algorithm is described in Section III. In Section IV, the experiments performed and the results of the experiments are presented. The paper concludes with a summary of the results in Section V.

II. Literature Review

A. Introduction to Genetic Algorithm

Genetic algorithm (GA) is a search algorithm that is inspired by the theory of genetics and natural selection [9], [10]. 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 [10] and Dumetrescu et al. [11] 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.

Repeat step 2 until a termination condition is met.

B. MCP Routing Algorithms

Various heuristics and approximation algorithms have been proposed to solve the MCP problem [12] – [25]. Other researchers turn their attentions to studying the network conditions that impact the performance of the routing algorithms developed [26] – [28]. Some researchers try to augment the developed algorithms by introducing other techniques such as smart forwarding [29] and pre-computation [30], [31].

Most of the QoS routing algorithms developed are actually a modified version of well-known shortest path routing algorithms such as Dijkstra's algorithm or Bellman-Ford algorithm [12] – [25]. There are also

approaches that are based on other methods such as flooding [46] and vector converting [47]. Another alternative approach would be to use artificial intelligence (AI) approach. Multi-constrained routing algorithms based on AI techniques such as fuzzy logic [48] and mobile agent [49] have been explored by researchers.

The earliest work concerning the MCP problem is an algorithm to solve two-constraint problem presented by Jaffe [12]. Jaffe uses an approach that combines the two link metrics into a single mixed link metric. The single link mixed metric is defined as follows:

$$w(e) = d_1 w_1(e) + d_2 w_2(e) \quad \text{(Eq. 1)}$$

where e is a link with two different metrics, $w_1(e)$ and $w_2(e)$ and d_1 and d_2 are positive real numbers. A feasible path is then searched by using Dijkstra's shortest path algorithm with respect to $w(e)$.

Chen and Nahrstedt [13] proposed an approximation to the MCP problem by scaling down $k - 1$ link metrics into integer metrics as follows:

$$w_i = \left\lceil \frac{w_i . x}{c_i} \right\rceil \quad \text{(Eq. 2)}$$

where x is a pre-defined positive integer and c_i is the constraint on metric w_i. The problem is then simplified to the problem of finding the shortest path with respect to a single QoS metric (the one that is not scaled down) with the condition that all the $k - 1$ scaled metrics are within the stricter constraint x_i.

Neve and Mighem proposed an algorithm called TAMCRA [14] to solve the general k-constrained algorithm. TAMCRA is based on three fundamental concepts: non-linear path length function, k-shortest path and the principle of non-dominated path. According to TAMCRA, the non-linear path length function can be approximated as follows:

$$w(p) = \max_{1 \le i \le k} \left(\frac{w_i(p)}{c_i} \right) \quad \text{(Eq. 3)}$$

where k is the number of constraints, $w(p)$ is the total path cost and c_i is the constraint on metric w_i. For each destination, TAMCRA stores n non-dominated paths. A path Q is said to be dominated by a path P if $w_i(P) \le w_i(Q)$ for all $i = 1, \ldots, k$.

Korkmaz and Krunz [17] extended the idea presented in TAMCRA to solve not only the MCP problem but also the more difficult multi-constrained optimization path (MCOP) problem where a primary cost function is to be minimized. The proposed algorithm is called H_MCOP. It uses the same approximated non-linear cost function as the one used by TAMCRA. It executes two modified version of Dijkstra's algorithm in backward and forward direction. In the backward direction, H_MCOP computes the shortest paths from every node to the destination node with respect to a linear combination of all the link metrics. In the forward direction, the non-linear cost function is used. Among all the feasible paths found, the algorithm would then find the one with the minimum primary cost function. If the problem to be solved is the MCP problem instead of the MCOP problem, then the algorithm can terminate once a feasible path is found.

Khadivi [25] improves upon the path length function used by TAMCRA and H_MCOP by introducing another component to the path length function. The function is defined as follows:

$$G(p) = \mu(p)\left[\Delta(p) + \varepsilon\right] \quad \text{(Eq. 4)}$$

where $\mu(p)$ and $\Delta(p)$ are defined as follows:

$$\mu(p) = \frac{1}{k} \sum_{i=1}^{k} \frac{w_i(p)}{c_i}$$

$$\Delta(p) = \sum_{i=1}^{k} \left(\frac{w_i(p)}{c_i} - \mu(p) \right)^2 \quad \text{(Eq. 5)}$$

where k is the number of constraints and c_i is the constraint on metric w_i. All of these algorithms assume that the network state information is correct and they return a feasible path if one is found. Even though TAMCRA can store n non-dominated paths, the algorithm does not check whether the paths stored are feasible or not.

A recent work by Guoliang Xue et al. [50] improves the method proposed by Chen and Nahrstedt [13], which resulted in a better run time complexity. However, the general concept is still the same where the shortest path is calculated based on only one QoS metric and the other $k - 1$ link metrics are approximated.

C. GA-based Routing Algorithms

Genetic algorithm (GA) is very good at solving optimization problems [10]. Since network routing is a form of optimization problem, many researchers have tried to apply GA to different types of network routing problems. Researchers in [32] – [37] address the shortest path routing problem, while the researchers in [38] – [43] address various types of QoS routing problems.

Chang [32] has applied the GA algorithm to the shortest path routing problem. He compared his result 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. Munetomo [33], [34] also applied GA to the shortest path routing problem. However, the objective of his algorithm is to generate alternative paths in addition to the path obtained from Dijkstra's algorithm. These alternative paths would allow the network to be more robust in the case of link or node failures. The alternate paths can quickly be used rather than having to compute a new path. Both Chang and Munetomo use variable length chromosome with each chromosome consisting of nodes that are on the path from sender to receiver. Other researchers who also use GA to solve the shortest path routing problem (or a variation of

it) are Sinclair [35], Shimamoto [36], and Hamdan [37].

Wang [38] and Riedl [39] develop a GA-based QoS routing algorithm to solve the delay-bandwidth-constraint routing problem. On top of finding a feasible path, these two algorithms are also designed to optimize network resource utilization. Wang's algorithm encodes the chromosome as a binary string where the genes represent the link between each pair of nodes in the network. Riedl uses chromosome that contains the weight of the links. Barolli [40], [41] tries to solve the problem of QoS routing subject to two QoS metrics which are delay and transmission success rate. According to [6], this problem is NP-complete. In these algorithms, the network is modeled as a tree and each gene in the chromosome represents a junction on the tree. Koyama [42] takes Barolli's work further by introducing a multi-purpose optimization method that would further improve its performance. Xiang [43] also proposes a GA-based QoS routing algorithm to solve a routing problem subject to four different QoS metrics: delay, bandwidth, loss-rate and jitter. The chromosome encoding used is similar to [38]. He [44] uses GA to solve the problem of route selection and capacity assignment, which is also NP-complete. Some of these algorithms are actually hybrid algorithms where GA is combined with another algorithm to produce better results, such as the ones in [35], [37] and [39].

Even though all the algorithms discussed above 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 in [32] – [34] use a list of node IDs from source to destination to represent the chromosome. Researchers in [38], [40] – [43] use a binary string to represent the chromosome. However, in the case of [40] – [42], the network is modeled as a tree and the binary string represents the junctions in the tree. In the case of [38] and [43], 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. The binary string comes from this representation. The crossover and mutation operations would depend on the genetic encoding used. The fitness function is the part of GA which is different for each algorithm. It is derived from the routing problem to be solved and the focus of the algorithm.

III. Proposal GA-based QoS Routing Algorithm

A. Algorithm Overview

The proposed algorithm is aimed to find *n* feasible paths given a set of QoS requirement. This is done to make the algorithm more robust in the case where the state information in the routers lags behind the network.

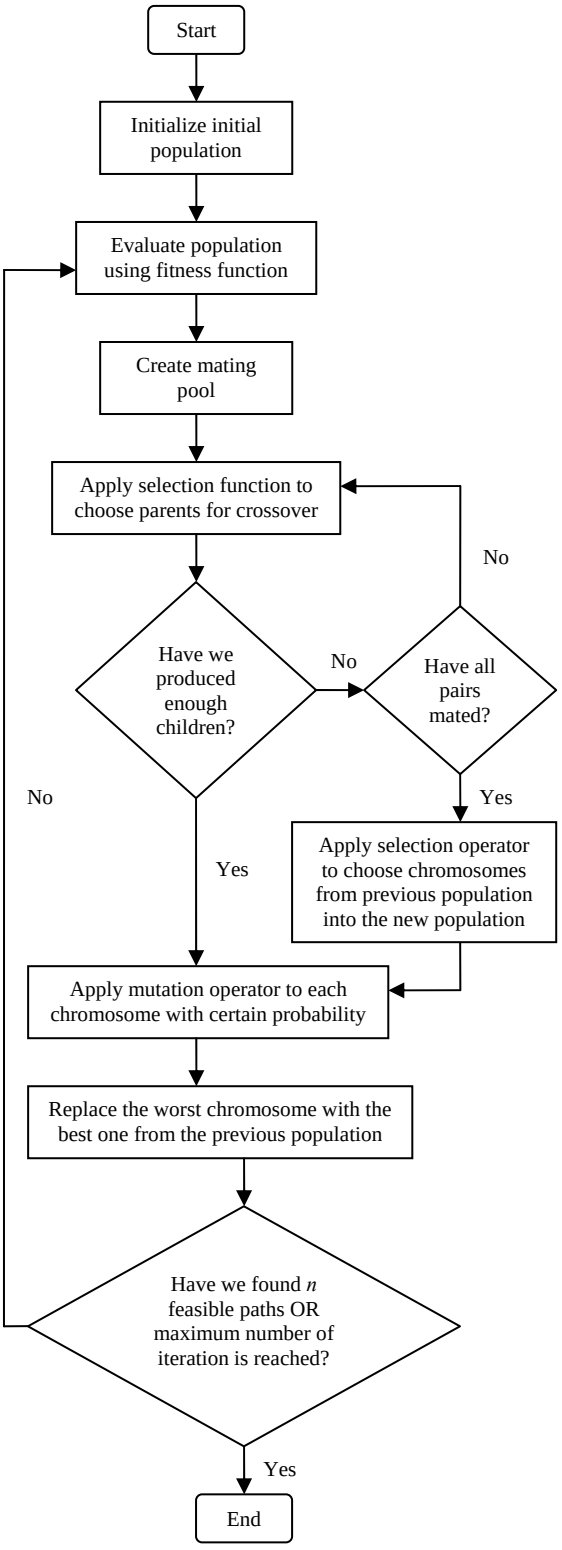

Figure 1: Overall procedure of the GA-based QoS routing algorithm

The idea is that if one of the feasible paths is no longer feasible due to the changes in the network state information, other alternate paths can be tried and hopefully one of them will turn out to be really feasible. In GA, multiple feasible paths can be easily generated by keep iterating the algorithm until *n* feasible paths are found. However, in order to prevent the algorithm from

running forever, the maximum number of iteration must be set.

As opposed to the algorithm proposed by Chang [32] and Munetomo [33], [34] where the objective of the algorithm is to optimize the link cost, 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 outline of the GA-based QoS routing 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 *n* feasible paths are found or the maximum number of iteration is reached.

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

B. 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 [6]. 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.

This GA-based routing algorithm is intended to be used in source routing where the sender performs 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 done. The first method is to follow the traditional GA where a chromosome is encoded as a string of binaries. This method is used by the algorithms in [38], [40] – [43]. The second method is to encode the chromosome using an encoding specific to the problem to be solved. This method is used by the algorithms in [32] – [34]. For our algorithm, the latter method is chosen. In our algorithm, 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].

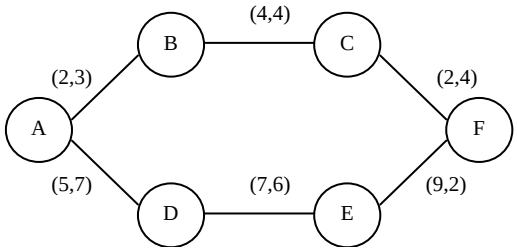

Figure 2: A small network with links that have two QoS metrics ($k = 2$)

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.

C. 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 to generate the random path is adapted from Chang [32]. 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.

D. 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 [11]. 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:

$$C_T = \sum_{k=0}^{K-1}\sum_{l=0}^{L-1} c_{kl} \quad \text{(Eq. 6)}$$

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:

$$c_{kl}' = \frac{c_{kl}}{\sum_{r=0}^{R-1}\sum_{j=0}^{L-1} c_{kj}} \quad \text{(Eq. 7)}$$

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 above, given a population of size p, the fitness function is defined as follows:

$$f_i = \frac{\sum_{j=0}^{K-1}\left[\frac{c_{ij}}{c_j}\right]}{K} \quad \text{(Eq. 8)}$$

where f_i represents the fitness value of the chromosome i, K represents the number of QoS metric on each 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:

$$f_0 = \frac{\frac{8}{29}+\frac{11}{26}}{2} = 0.35 \quad f_1 = \frac{\frac{21}{29}+\frac{15}{26}}{2} = 0.65$$

In the example above, 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.

E. 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 [11], 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.

In this algorithm, the pairwise tournament selection with tournament size, s = 2, is employed. 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. A pair of parent chromosomes can only be selected once.

F. 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 [11]. 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 (i.e. crossover probability, p_c = 1). The actual crossover operation is similar to the one described in [32] – [34]. 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 [32]. 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.

G. Mutation

The objective of mutation operation is to create diversity in the population [11]. 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 Chang [32]. 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 ith new node in the path. The new path is generated randomly; the same way as the paths in the initial population is generated.

IV. Experiments and Discussion

The proposed algorithm has been implemented as a C++ program and is run on a computer with 2.4GHz Core 2 processor and 4GB of RAM. The performance of the algorithm is compared with TAMCRA [14], Chen [13], H_MCOP [17] and Khadivi [25] algorithms. The TAMCRA algorithm used is configured to store two non-dominated paths. The performance metrics used to compare the algorithms are success ratio (SR) and feasibility ratio (FR). The success ratio is defined as the number of feasible path found using the node's state information divided by the total number of requests. Feasibility ratio is defined as the number of paths that is actually feasible given that the current state information has changed divided by the total number of requests.

There are two types of network used in the simulation, the *n x n* mesh network and the Waxman network [45]. 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:

$$p_{ij} = \alpha \exp(-(\frac{d_{i,j}}{\beta L})),\ 0 < \alpha, \beta < 1 \quad \text{(Eq. 9)}$$

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 small value of β increases the density of

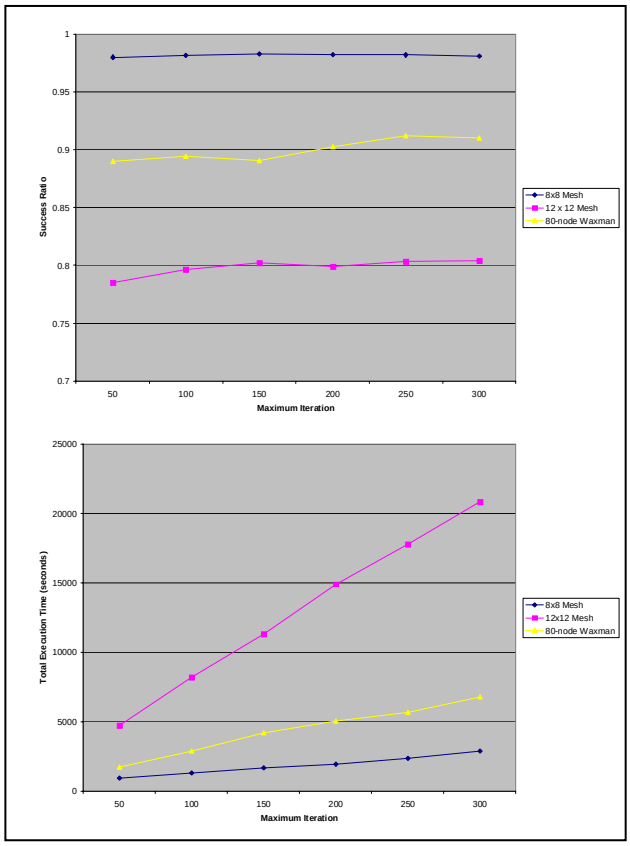

Figure 3: Success ratio and execution time with respect to maximum number of iterations

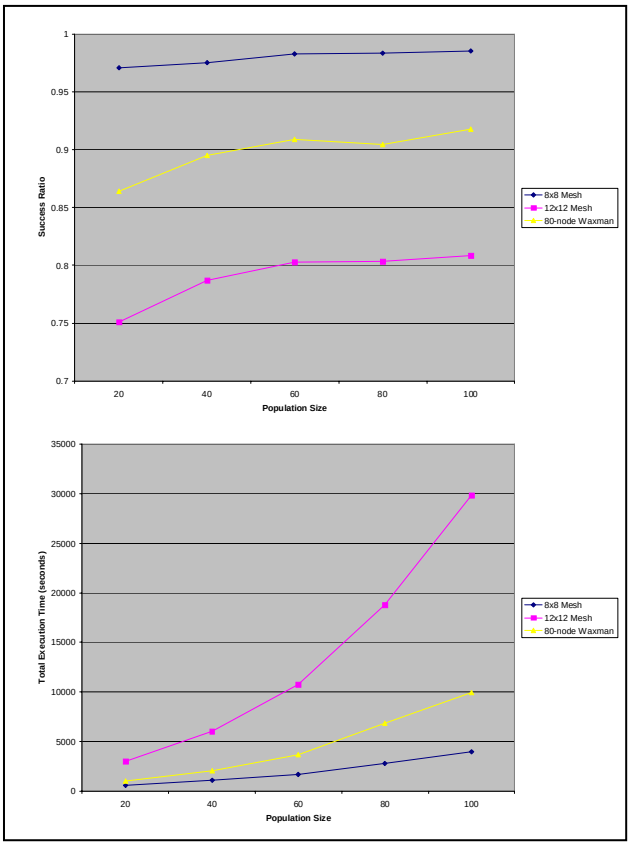

Figure 4: Success ratio and execution time with respect to population size

short edges relative to longer ones. In all the experiments, the values for α and β are set to 0.2 and 0.1 respectively. However, to avoid having isolated nodes, each node must be connected to at least one other node. Each link in the graph is associated with k randomly generated link metrics, where each link metric is given a value $w_k(i,j) \sim uniform[1,100]$. The source and destination nodes are also generated randomly. For each QoS requests, the constraints for each metric are randomly generated and is given a value $c_k \sim uniform[400,600]$ for the mesh network and $c_k \sim uniform[200,400]$ for the Waxman network.

Each node is assumed to have the same set of network state information. All the routing algorithms will use the node's state information to find a feasible path. However, before data is actually sent, each metric of each link will have a small probability, p_v, to be increased or decreased by a random value $v_k \sim uniform[0,max\_change]$. The success of data transmission will now be determined based on the new link metrics.

Each simulation is run on 8 x 8 mesh network, 12 x 12 mesh network and 80-node Waxman networks. The results reported in the subsequent sections are 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 new topology with different connectivity is generated for each run. Therefore, the result for 80-node Waxman networks is actually obtained from 50 different, randomly generated topologies. In each run, a total of 5000 connection requests are generated where the source and destination nodes are chosen randomly.

In short, each single point in the graphs presented in subsequent sections is obtained by taking the grand average of the results produced by the algorithm after running it 250,000 times (50 runs multiply with 5000 connection requests). This gives the average performance of the algorithm when run on a network of the same size, but subjected to different set of QoS metrics on each link, different nodes connectivity (in the case of Waxman networks), various source-destination pair and various QoS requirement.

A. Identifying the Proper GA Parameters

In GA, there are several parameter values that need to be decided during implementation. For the proposed algorithm, the values for maximum iteration, population size and mutation probability need to be chosen properly to in order to ensure optimal performance. The values for maximum iteration and population size allows for a tradeoff between the quality of the result and the computation time. A higher maximum iteration and population size would, in general, gives a better quality result at the expense of higher computation time. Mutation probability on the other hand must be chosen correctly because any value that is too low or too high would cause the algorithm to perform badly.

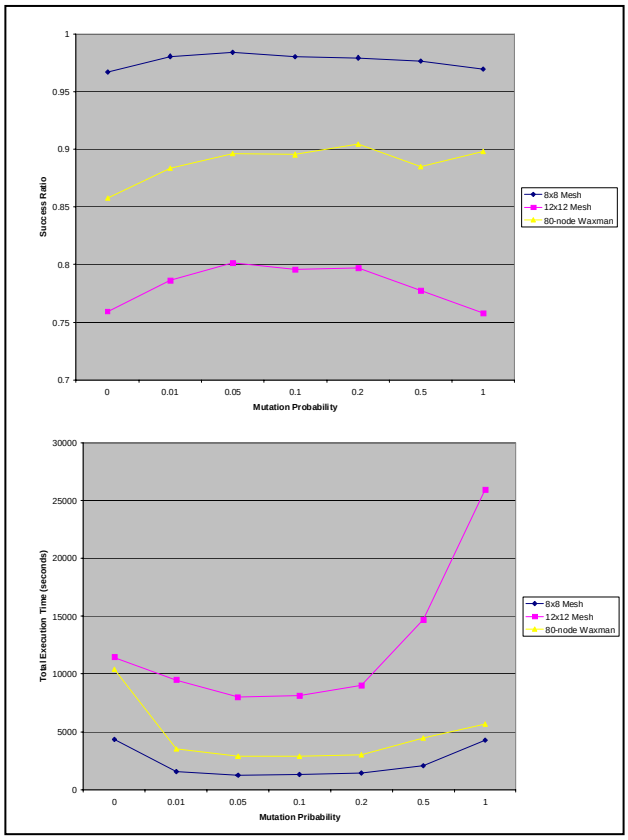

Figure 5: Success ratio and execution time with respect to mutation probability

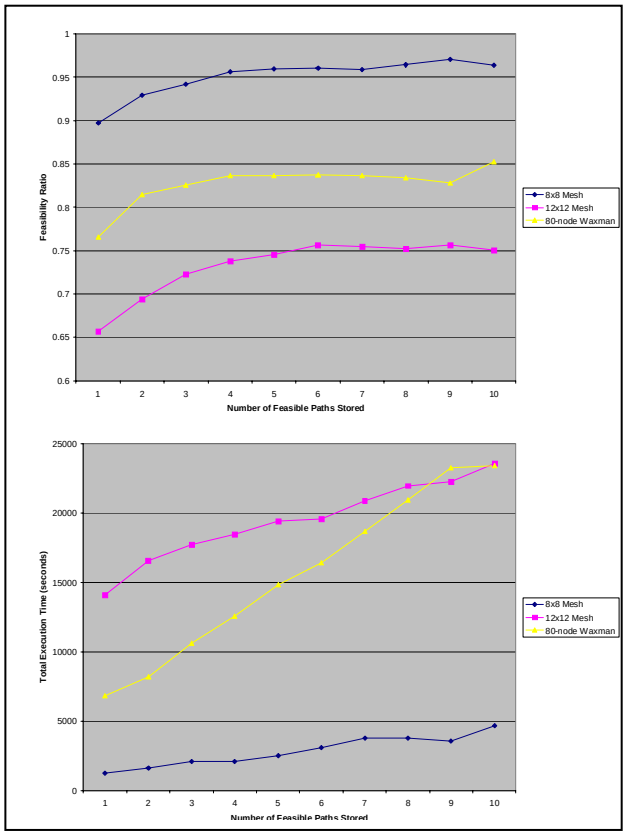

Figure 6: Feasibility ratio and execution time with respect to the number of feasible path generated

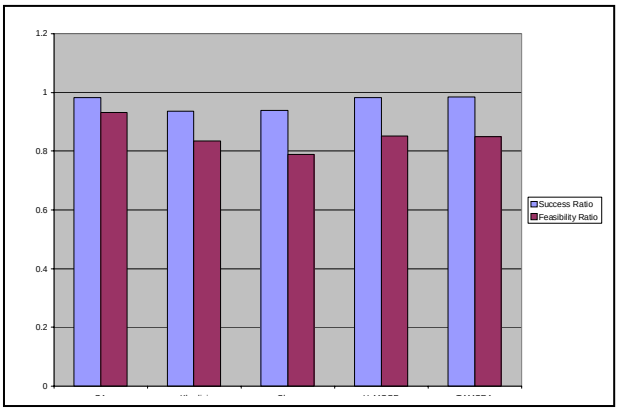

Figure 7: Success ratio and feasibility ratio for 8x8 mesh network

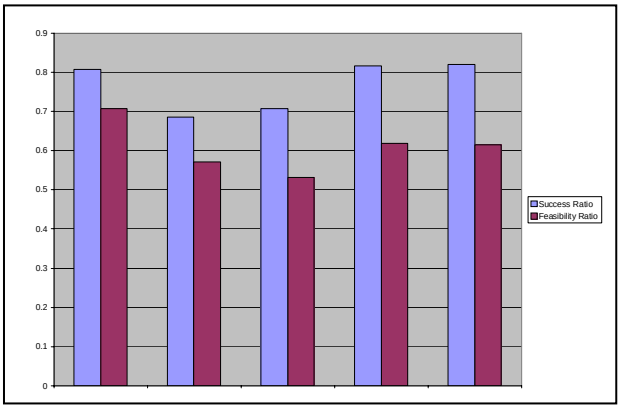

Figure 8: Success ratio and feasibility ratio for 12x12 mesh network

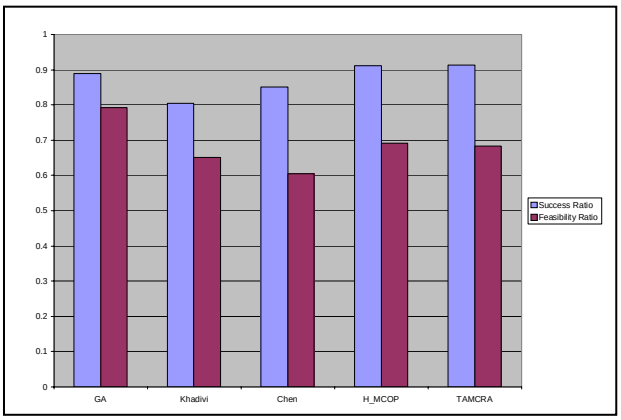

Figure 9: Success ratio and feasibility ratio for 80-node Waxman networks

Here, the effect of the different GA parameters will be tested experimentally. For each of the parameters, the simulation is run several times with different values of the intended parameter, while fixing the values of the other two. The performance of the algorithm with respect to the different parameter values is evaluated using the success ratio and computation time. For all the simulations in this experiment, the number of QoS metrics on each link is set to two ($k = 2$).

Figure 3, Figure 4 and Figure 5 show the success ratio and execution time for different values of maximum iteration, population size and mutation probability respectively. For maximum iteration, although in general

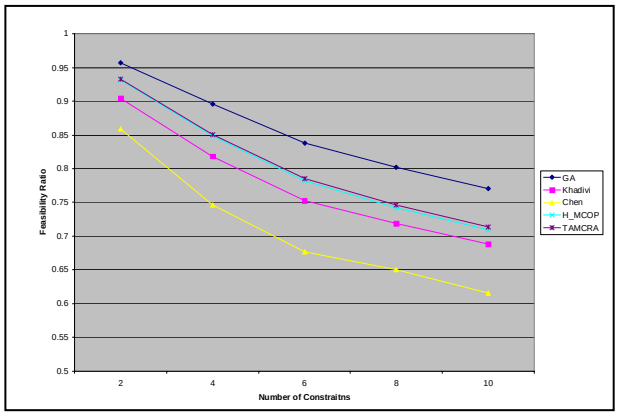

Figure 10: Feasibility ratio with respect to number of constraints for 8x8 mesh network

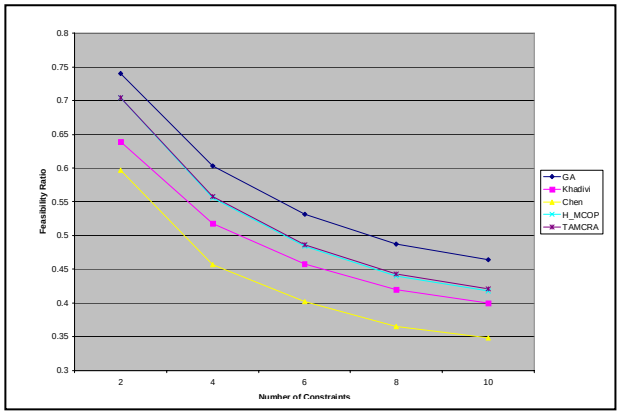

Figure 11: Feasibility ratio with respect to number of constraints for 12x12 mesh network

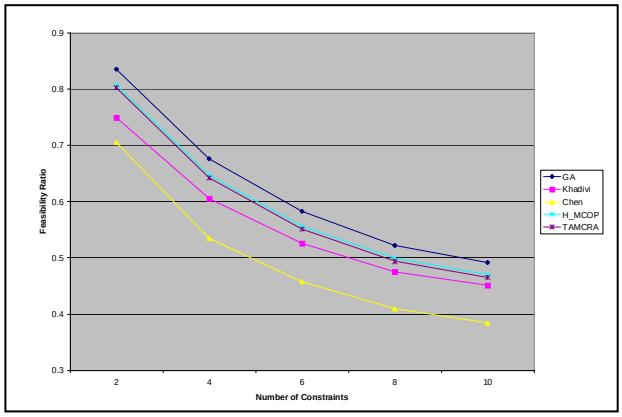

Figure 12: Feasibility ratio with respect to number of constraints for 80-node Waxman networks

the success ratio tends to be higher with higher value of maximum iteration, the increase in success ratio is not consistent. There is no guarantee that a higher success ratio will be achieved with a higher maximum iteration. Even if there is an increase in success ratio, the increase is not very significant. This behavior shows that if a feasible path exists, most of the time it can be found without requiring a large number of iterations. As such, an increase in the maximum iteration value does not significantly increase the success ratio. On the other hand, the total execution time increases linearly with higher value of maximum iteration. With respect to population size, the success ratio tends to increase with higher population size. However, the increase is logarithmic, meaning that the rate of increase gets lower as the population size gets larger. This trend is the same regardless of the network size. The execution time, however, increases exponentially. For mutation probability, there seems to be a positive correlation between the success ratio and execution time where the mutation probability that gives a good success ratio also tends to give a low execution time. Although the right value for mutation probability varies depending on the network topology, it can be generalized that any value between 0.05 and 0.2 tends to give a good result with low execution time.

For the rest of the experiment, the values for maximum iteration, population size and mutation probability are chosen to be 200, 60 and 0.1 respectively.

B. Generating Multiple Feasible Paths

To ensure the robustness of the proposed algorithm with respect to changing link cost values, the algorithm can be configured to return multiple feasible paths. This is done by keep iterating the algorithm until the specified n feasible paths have been found. However, the maximum number of iteration performed is still subjected to the maximum iteration parameter as discussed above. As such, it is possible that the number of feasible paths returned is less than n, especially if the actual number of feasible path is very low. In this experiment, the performance of the algorithm will be evaluated with respect to the values of n. The performance metric used is the feasibility ratio. For all the simulations in this experiment, the number of QoS metrics on each link is also set to two (k = 2). It is expected that the feasibility ratio will increase as the value of n increases. However, since there is a limit on the actual number of feasible path and the value of maximum iteration, there should be a maximum values of n after which a higher value of n would no longer be useful.

This experiment is run for n ranging from 1 to 10. To measure feasibility ratio, the link metrics must be able to change before a packet is transmitted. As explained above, there are two variables that control how the link metrics change. These two variables are probability of link metric change, p_v, and maximum link metric change offset, *max_change*. For this experiment, the values for p_v and *max_change* are set to 0.2 and 50 respectively.

Figure 6 shows the feasibility ratio and execution time with respect to different values of n. The result shows that the feasibility ratio increases logarithmically as more feasible paths are generated. The increase is more significant in the mesh networks as compared to Waxman networks. In Waxman networks, the feasibility ratio can only improve up to n = 3. Storing more feasible paths than three does not increase the performance of the algorithm. This can be explained by the fact that there are not many alternative paths between two nodes in Waxman networks as compared to mesh networks. As a result, increasing the number of generated paths will not have much effect on the performance of the algorithm.

The execution time increases linearly as more feasible paths are generated. However, the execution time in Waxman networks tends to have a higher rate of increase as compared to the rate of increase in the mesh networks.

C. Comparison with Other MCP Algorithms

In this experiment, the performance of the proposed GA-based QoS routing algorithm will be compared with several of the other MCP algorithms discussed in Section II above. For this simulation, the algorithm will be executed using the state information known to all the routing nodes. From this, the success ratio given by each algorithm will be recorded. Before the packet is sent, some of the links will have their link metrics changed, subjected to the value of p_v and *max_change*. For this experiment, the value of p_v and *max_change* are set to 0.2 and 100 respectively. The feasibility ratio given by each algorithm would then be recorded. Each link in the network is associated with two QoS metrics (k = 2) and the number of feasible paths stored is five (n = 5).

Figure 7 until Figure 9 show the success ratio and the feasibility ratio for the three types of network used in this experiment. For the 8x8 mesh network, the success ratio given by the proposed algorithm is comparable to the ones given by the other algorithms. However, the advantage of the proposed algorithm becomes more apparent when it comes to feasibility ratio where it gives the highest feasibility ratio compared to the other algorithms. In the 12x12 mesh network and 80-node Waxman networks, the proposed algorithms starts out with a lower success ratio compared to some of the other algorithms. But when it comes to feasibility ratio, it still gives the highest. From this result, it can be seen that the proposed algorithm is almost comparable to the other MCP algorithms in terms of success ratio. However, when used in an unstable network environment where the link metrics can easily change, the proposed algorithm has shown its ability to be more robust and give a higher feasibility ratio compared to the other MCP algorithms.

D. Scalability with Respect to Number of Constraints

Since the proposed algorithm is intended to solve the general k-constrained problem, we will next test the performance of the proposed algorithm with different number of constraints. The performance metric used here is the feasibility ratio. For this experiment, the value of p_v and *max_change* are set to 0.2 and 50 respectively and the number of feasible paths stored is set to five (n = 5).

Figure 10 until Figure 12 show the feasibility ratio given by the different algorithms as the number of constraints increases from 2 to 10. For the 8x8 and 12x12 mesh networks, the proposed algorithm gives the highest feasibility ratio regardless of the number of constraints. However, as the number of constraints gets larger, the difference between the feasibility ratio of the proposed algorithm and the feasibility ratio of the other algorithms also becomes larger. This trend is also similar in 80-node Waxman networks although in this case, the feasibility ratio given by the proposed algorithm starts out slightly lower compared to the feasibility ratio of TAMCRA and H_MCOP when the number of constraints is low. However, as the number of constraints gets larger, the proposed algorithm starts to give a better feasibility ratio. From this result, it can be implied that the proposed algorithm can scale better with larger number of constraints as compared to the other MCP algorithms and therefore it is suitable to be used for the general k-constrained MCP routing problem.

V. Conclusion

This paper proposed a robust GA-based QoS routing algorithm for the general k-constrained MCP routing problem. As opposed to most other MCP routing algorithms, the proposed algorithm is able to produce multiple feasible paths and therefore can perform well in a network environment where the state information in the router lags behind the network. Based on the experiments performed, the performance of the algorithm is comparable to the other MCP routing algorithms in terms of success ratio but consistently outperform them in terms of feasibility ratio. The algorithm is also shown to be more scalable compared to the other MCP routing algorithms with respect to the number of link constraints. The advantage of the algorithm is more apparent in n x n mesh networks as compared to Waxman networks due to the availability of large number of alternative paths between any two nodes in the mesh networks.

References

[1] M. F. Daneshmand, R. R. Roy, C. G. Savolaine, "Framework and requirements of quality of service for multimedia applications", *Intelligent Information System*, pp. 466 – 474, December 1997.

[2] D. Ghosh, V. Sarangan, R. Acharya, "Quality-of-service routing in IP networks", *IEEE Transactions on Multimedia*, vol. 3, issue 2, pp. 200 – 208, June 2001.

[3] Shigang Chen, K. Nahrstedt, "An overview of quality of service routing for next-generation high-speed networks: problems and solutions", *IEEE Network*, vol. 12, issue 6, pp. 64 – 79, November / December 1998.

[4] G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda, T. Przygienda, "QoS Routing Mechanisms and OSPF Extensions", *RFC 2676*, August 1999.

[5] R. Guerin, A. Orda, "QoS-based routing in networks with inaccurate information: theory and algorithms", *IEEE/ACM Transactions in Networking*, vol. 7, pp. 350 – 364, June 1999.

[6] Zheng Wang, J. Crowcroft, "Quality-of-service routing for supporting multimedia applications", *IEEE Journals on Selected Areas in Communications*, vol. 14, issue 7, pp. 1228 – 1234, September 1996.

[7] Shigang Chen, K. Nahrstedt, "Distributed quality-of-service routing in high-speed networks based on selective probing", *23rd Annual Conference on Local Computer Networks*, pp. 80 – 89, October 1998.

[8] Yi Yang, J. K. Muppala, S. T. Chanson, "Quality of service routing algorithms for bandwidth-delay constrained applications", *9th International Conference on Network Protocols*, pp. 62 – 70, November, 2001.

[9] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[10] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.

[11] D. Dumetrescu, B. Lazzerini, L.C. Jain, A. Dumitrescu, *Evolutionary Computing*. The CRC Press, 2000.
[12] J. M. Jaffe, "Algorithms for finding paths with multiple constraints", *Networks*, vol. 14, pp. 95 – 116, 1984.
[13] S. Chen, K. Nahrstedt, "On finding multi-constrained path", *IEEE International Conference on Communication*, vol. 2, pp. 874 – 879, June 1998.
[14] H. De Neve, P. Van Mieghem, "A multiple quality of service routing algorithm for PNNI", *IEEE ATM Workshop*, pp. 324 – 328, May 1998.
[15] L. L. H. Andrew, A. A. N. Ananda Kusuma, "Generalized analysis of a QoS-aware routing algorithm", *IEEE Global Telecommunications Conference*, vol. 1, pp. 1 – 6, November 1998.
[16] Jun Song, Hung Keng Pung, L. Jacob, "A multi-constrained distributed QoS routing algorithm", *IEEE International Conference on Networks*, pp. 165 – 171, September 2000.
[17] T. Kormaz, M. Krunz, "Multi-constrained optimal path selection", *20th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 834 – 843, April 2001.
[18] Dong-Won Shin, E. K. P. Chong, H. J. Siegel, "A multiconstraint QoS routing scheme using the depth-first search method", *IEEE Workshop on High Performance Switching and Routing*, pp. 385 – 389, May 2001.
[19] Gang Feng, K. Makki, N. Pissinou, C. Douligeris, "An efficient approximate algorithm or delay-cost-constrained QoS routing", *10th International Conference on Computer Communications and Networks*, pp. 395 – 400, October 2001.
[20] F. Kuipers, P. Van Mieghem, T. Korkmaz, M. Krunz, "An overview of constraint-based path selection algorithms for QoS routing", *IEEE Communications Magazine*, vol. 40, issue 12, pp. 50 – 55, December 2002.
[21] Xin Yuan, "Heuristics algorithms for multiconstrained quality-of-service routing", *IEEE/ACM Transactions on Networking*, vol. 10, issue 2, pp. 244 – 256, April 2002.
[22] Guoliang Xue, Arunahba Sen, Rakesh Banka, "Routing with many additive QoS constraints", *IEEE International Conference on Communications*, vol. 1, pp. 223 – 227, May 2003
[23] Baoxian Zhang, H. T. Mouftah, "A stateless QoS routing algorithm subject to multiple constraints", *IEEE International Conference on Communications*, vol. 3, pp. 1870 – 1874, May 2003.
[24] R. A. Resende, R. M. Oliveira, F. J. L. Padua, Akebo Yamakami, I. S. Bonathi, A. C. Lavelha, "An algorithm for quality-of-service unicast routing in data communication networks", *13th International Conference on Telecommunications*, May 2006.
[25] P. Khadivi, S. Samavi, T. D. Todd, H. Saidi, "Multi-constraint QoS routing using a new single mixed metric", *IEEE International Conference on Communications*, vol. 4, pp. 2042 – 2046, June 2004.
[26] F.A. Kuipers, P. Van Mieghem, "The impact of correlated link weights on QoS routing", *22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 1425 – 1434, March / April 2003.
[27] F. A. Kuipers, P. F. A Van Mieghem, "Conditions that impact the complexity of QoS routing", *IEEE/ACM Transactions on Networking*, vol. 13, issue 14, pp. 717 – 730, August 2005.
[28] B. Peng, A. H. Kemp, S. Boussakta, "Impact of network conditions on QoS routing algorithms", *3rd IEEE Consumer Communications and Networking Conference*, vol. 1, pp. 25 – 29, January 2006.
[29] A. Fei, M. Gerla, "Smart forwarding technique for routing with multiple QoS constraints", *IEEE Global Telecommunication Conference*, vol. 1, pp. 599 – 604, November / December 2000.
[30] Yong Cui, Ke Xu, Jianping Wu, "Precomputation for multi-constrained QoS routing in high-speed networks", *22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 1414 – 1424, March / April 2003.
[31] Won-Ick Lee, Byeong Gi Lee, "Multi-constrained pre-computation based selective probing (MC-PCSP) scheme for distributed QoS routing", *Joint Conference of the 10th Asia-Pacific Conference on Communications and the 5th International Symposium on Multi-Dimensional Mobile Communications*, vol. 1, pp. 228 – 233, August / September 2004.
[32] Chang Wook Ahn, R.S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations", *IEEE Transactions on Evolutionary Computing*, vol. 6, pp. 566 – 579, December 2002.
[33] M. Munetomo, Y. Takai, Y. Sato, "A migration scheme for the genetic adaptive routing algorithm", *IEEE International Conference on Systems, Man and Cybernatics*, vol. 3, pp. 2774 – 2779, October 1998.
[34] M. Munetomo, N. Yamaguchi, K. Akama, Y. Sato, "Empirical investigations on the genetic adaptive algorithm in the Internet", *2001 Congress of Evolutionary Computing*, vol. 2, pp. 1236 – 1243, May 2001.
[35] M. C. Sinclair, "Minimum cost routing and wavelength allocation using genetic algorithm / heuristic hybrid approach", *6th IEE Conference on Telecommunications*, pp. 62 – 71, March 1998.
[36] N. Shimamoto, A. Hiramatsu, K. Yamasaki, "A dynamic routing control based on a genetic algorithm", *IEEE Conference in Neural Networks*, pp. 1123 – 1128, March 1993.
[37] M. Hamdan, M.E. El-Hawary, "Hopfield-Genetic approach for solving the routing problem in computer networks", *Canadian Conference on Electrical and Computer Engineering*, vol. 2, pp. 823 – 827, May 2002.
[38] Wang Xinhong, Wang Guangxing, "An algorithm for QoS routing to optimize network resource utilization", *International Conference on Info-tech and Info-net*, vol. 2, pp. 474 – 479, October / November 2001.
[39] A. Riedl, "A hybrid genetic algorithm for routing optimization in IP networks utilizing bandwidth and delay metrics", *IEEE Workshop on IP Operations and Management*, pp. 166 – 170, 2002.
[40] L. Barolli, A. Koyama, H. Sawada, T. Suganuma, N. Shiratori, "A new QoS routing approach for multimedia applications based on genetic algorithm", *First International Symposium on Cyber Worlds*, pp. 289 – 295, November 2002.
[41] L. Barolli, A. Koyama, K. Matsumoto, T. Suganuma, N. Shiratori, "A genetic algorithm based routing method using two QoS parameters", *13th International Workshop on Database and Expert Systems Applications*, pp. 3 – 7, September 2002.
[42] A. Koyama, L. Barolli, K. Matsumoto, B. O. Apduhan, "A GA-based multi-purpose optimization algorithm for QoS routing", *18th International Conference on Advanced Information Networking and Applications*, vol. 1, pp. 23 – 28, 2004.
[43] F. Xiang, L. Junzhou, W. Jieyi, G. Guanqun, "QoS routing based on genetic algorithm", *Computer Communications*, vol. 22, issue 15 – 16, pp. 1392 – 1399, September 1999.

[44] He Cuihong, "Route selection and capacity assignment in computer communication networks based on genetic algorithm", *IEEE International Conference on Intelligent Processing Systems*, vol. 1, pp. 548 – 552, October 1997.
[45] B. M. Waxman, "Routing for multipoint connections", *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. December 1988.
[46] Y.-S. Yen, R.-S. Chang, H.-C. Chao, "Flooding limited for multi-constrained quality-of-service routing protocol in mobile ad hoc networks", *IET Communications*, Vol. 2, Issue 7, pp. 971 – 981, August 2008.
[47] Fu-Sheng Dai, Ai-Jun Liu, "A multi-constrained quality-of-service routing algorithm based on vector converting", *5th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1 – 4, September 2009.
[48] Zuo Jing, Chi Xuefen, Lin Guan, Li Hongxia, "Service-aware multi-constrained routing protocol with QoS guarantee based on fuzzy logic", *22nd International Conference on Advanced Information Networking and Applications Workshop*, pp. 762 – 767, March 2008.
[49] Chen Wei, Zhang Yi, "A multi-constrained routing algorithm based on mobile agent for MANET networks", *International Joint Conference on Artificial Intelligence*, pp. 16 – 19, April 2009.
[50] Guoliang Xue, Weiyi Zhang, Jian Tang, K. Thulasiraman, "Polynomial time approximation algorithms for multi-constrained QoS routing", *IEEE/ACM Transactions on Networking*, Vol. 16, Issue 3, pp. 656 – 669, June 2008.

Salman Yussof was born in Kuala Lumpur, Malaysia on 27 October 1976. He received his Bachelor of Science degree and Masters of Science degree in Electrical and Computer Engineering from Carnegie Mellon University, USA, in 1999.

In the same year, he was accepted as a faculty member at the College of Information Technology, Universiti Tenaga Nasional, Malaysia and is currently a Senior Lecturer at the same institution. His research interests include quality-of-service (QoS) provisioning in computer networks, application of artificial intelligence (AI) to network problems, parallel computing and ad-hoc networks.

Mr. Salman is a member of Association for Computing Machinery (ACM), Boards of Engineers Malaysia (BEM) and Malaysian National Computer Federation (MNCC).

Ong Hang See is an Associate Professor at the College of Engineering, Universiti Tenaga Nasional, Malaysia. He received his Bachelor of Science degree in Electrical Engineering and Master of Science degree in Health Physics from University of North Dakota, USA, in 1986 and 1989, respectively. Later, he was educated and trained in University of Minnesota where he received his Ph.D in Biomedical Engineering and Medical Physics.

His job experience includes title like Scientific and Database Programmer, Network Administrator and Designer, Biomedical Engineer, and Instrumental Specialist. He has designed and implemented instrument such as high field nonhuman Magnetic Resonance Spectroscopy and Imaging system, networked data acquisition system and ultrasonic power meter. In networking field, he was involved in designing and testing of an ATM network for healthcare education and telemedicine. Among the renowned biomedical research institution that he worked, attached and collaborated are United States Department of Agriculture (USDA), Grand Forks Human Nutrition Research Center, University of Minnesota Hospital & Clinics (UMHC), Center for Magnetic Resonance Research (CMRR), Center for Interdisciplinary Applications in Magnetic Resonance (CIAMR), and Mayo Clinics. Currently his research interests include internet technology, biological computing, biomedical imaging and instrumentation.

Dr. Ong is a long time member of the Institute of Electrical and Electronics Engineers (IEEE). He has also served as a member of the Malaysian Radiology Society Teleradiology Standard Committee.