Applying GA to Find Minimum Cost Routes between Existing Nodes in Different Network Models

Ahmad Al-Sayaad, Abeer Abu-Sultan, Zakaria Zayed, Deena Abu-Sultan

AXIS Solutions
P. O. Box 64418 Shuwaikh B, 70455, Kuwait
Ahmad.Al-Sayaad@axis-solutions.com
Abeer.Abu-Sultan@axis-solutions.com
Zakaria.Zayed@axis-solutions.com
Deena.Abu-Sultan@axis-solutions.com

***Abstract*— A Genetic Algorithm based techniques is introduced for finding a solution for the problem of finding minimum cost routes within a network. GA approach is selected for solving such problems due to its adaptive nature where possible solutions are integrated to produce better solutions until reaching the optimal solution. The implemented GA is used to find the minimum cost routes between nodes of different networks models. Three network models are introduced and analyzed: Random, Small-World and Scale-Free network models. Each network model has its own structure and properties. The overall fitness value for each network, which represents the average fitness value for the minimum cost routes within the network, is calculated and compared for the studied network models. The produced experiments provide reasonable results in terms of minimum cost routes within a network, which might be used by network designers and administrators to suggest network redesign techniques in order to overcome routing bottlenecks within any given network.**

***Keywords*— GA, Average fitness, Minimum cost route, Complex networks, Average network degree.**

I. INTRODUCTION

We are going to use Genetic Algorithms (GA) to develop an algorithm for finding the minimum cost routes between any selected pair of nodes within different network models. First of all, finding the minimum cost route between two nodes within a network is an optimization problem. Therefore, using GA approach could be suitable for solving such problems due to its adaptive nature where possible solutions are integrated to produce better solutions until reaching the optimal solution. GA is based on few known steps: Generation, Crossover and mutation. Its implementation requires setting and defining design constraints, and changing the algorithm to best fit the problem solving approach. The optimal solution is a baseline in the performance evaluation criteria; hence network administrators and designers can use this information toward redesigning the network to get a better structure with higher throughput.

The topology of the network plays major role in determining the optimal route between a pair of nodes. Searching for the optimal network topology in which we identify two nodes of minimum cost is not a trivial task. Here, we intend to examine several network structures and find the optimal solutions. Evidently, there exists no optimal network topology satisfying our objective function. Consequently, generating several network structures and optimize each one of them. Three main networks structures are examined: random, small world and scale-free.

In this research, we demonstrate the application of GA on finding the optimal solution in all of the three network models. We are going to find the minimum cost routes between any selected nodes in different networks models using a GA approach. Then we are going to evaluate different network models in terms of existing minimum cost routes within their structure.

II. GENETIC ALGORITHM (GA)

A genetic algorithm (GA) is a programming technique that mimics biological evolution as a problem-solving strategy. Given a specific problem to solve, the input to the GA is a set of potential solutions to that problem, encoded in some fashion, and a metric called a *fitness function* that allows each candidate to be quantitatively evaluated. These candidates may be solutions already known to work, with the aim of the GA being to improve them, but more often, they are generated at random.

The GA then evaluates each candidate according to the fitness function. In a pool of randomly generated candidates, of course, most will not work at all, and these will be deleted. However, purely by chance, a few may hold promise - they may show activity, even if only weak and imperfect activity, toward solving the problem. These promising candidates are kept and allowed to reproduce. Multiple copies are made of them, but the copies are not perfect; random changes are introduced during the copying process. These digital offsprings then go on to the next generation, forming a new pool of candidate solutions, and are subjected to a second round of fitness evaluation. Those candidate solutions which were worsened, or made no better, by the changes to their code are again deleted; but again, it is possible that the random variations introduced into the population may have improved some individuals, making them into better, more complete or more efficient solutions to the problem at hand. These winning individuals are selected and copied over into the next generation with random changes, and the process repeats. The expectation is that the average fitness of the population will increase each round, and so by repeating this process for hundreds or thousands of rounds, very good solutions to the problem can be discovered.

As astonishing and counterintuitive as it may seem to some, genetic algorithms have proven to be an enormously powerful and successful problem-solving strategy, dramatically demonstrating the power of evolutionary principles. Genetic algorithms have been used in a wide variety of fields to evolve solutions to problems as difficult as or more difficult than those faced by human designers. Moreover, the solutions they come up with are often more efficient, more elegant, or more complex than anything comparable a human engineer would produce [1].

Any implemented GA is made of few known steps. These steps might slightly vary from one implementation to another. However, the main steps must be implemented in order to have a meaningful genetic algorithm. In general, the main steps necessary to complete any given genetic algorithm are presented below in Fig. 1.

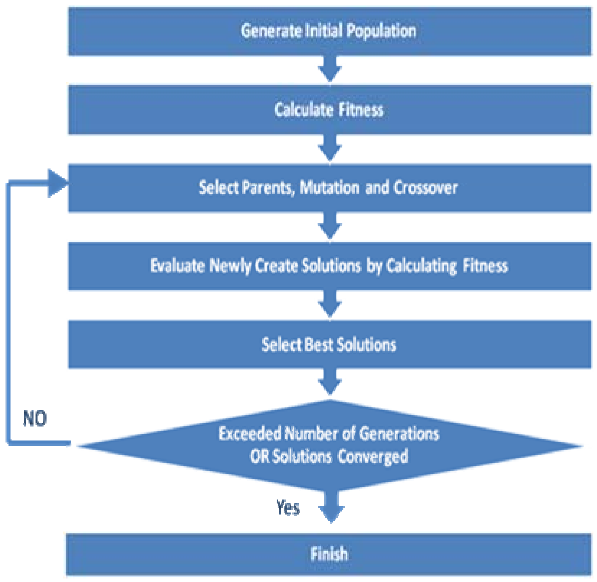
Generate Initial Population
Calculate Fitness
Select Parents, Mutation and Crossover
Evaluate Newly Create Solutions by Calculating Fitness
Select Best Solutions
NO
Exceeded Number of Generations OR Solutions Converged
Yes
Finish

Fig. 1 Main steps used by any given Genetic Algorithm

In the following sections, parameters constraints and requirements along with detail description of the involved steps for a given GA are going to be presented in more details to provide better understanding of the design and implementation of a given GA.

A. GA Constraints and Parameters

Implementing GA requires defining the constraints and setting some parameters that are going to be used within the algorithm toward solving the optimization problem. These constraints and parameters vary from how to represent a single solution, to the way that individual solutions are selected and kept for later generations. Below, we are going to represent the basic constraints and parameters used while implementing GA.

1) Representation: Before a genetic algorithm can be put to work on any problem, a method is needed to encode potential solutions to that problem in a form that a computer can process. One common approach is to encode solutions as binary strings: sequences of 1's and 0's, where the digit at each position represents the value of some aspect of the solution. Another, similar approach is to encode solutions as arrays of integers or decimal numbers, with each position again representing some particular aspect of the solution. This approach allows for greater precision and complexity than the comparatively restricted method of using binary numbers only and often is intuitively closer to the problem space [2].

2) *Initial Population:* The initial population is a set of randomly generated solutions for a given problem. This initial population is created randomly in order to cover different variations of the solution and avoid early convergence when applying GA. The size of the initial population is specified by the user and it must be large enough to provide better sample space of the possible solutions for a given problem. The initial population is then used as the basis for the applied GA and therefore, all later generations are made of this population.

3) Fitness Function: A fitness function is a particular type of objective function that prescribes the optimality of a solution (that is, a chromosome) in a genetic algorithm so that that particular chromosome may be ranked against all the other chromosomes. Optimal chromosomes, or at least chromosomes which are more optimal, are allowed to breed and mix their datasets by any of several techniques, producing a new generation that will (hopefully) be even better. An ideal fitness function correlates closely with the algorithm's goal, and yet may be computed quickly. Speed of execution is very important, as a typical genetic algorithm must be iterated many, many times in order to produce a usable result for a non-trivial problem [3].

4) *Selection, Mutation and Crossover:* Selection, mutation and crossover are used together to regenerate a new population that possibly have better overall fitness. Selection defines the criterion that is used to select the best solutions and keep them for next generation. For example, a selection technique is to keep the solutions with the highest fitness for next generation and replace the worst ones with newly generated offspring. On the other hand, mutation and crossover are methods used to generate new solutions based on existing chromosomes. Mutation is based on randomly changing a small portion of the original selected parent to generate a new child solution. Crossover is done by selecting two parent solutions and making a crossover between them to get new children solution. Crossover could be implemented in different ways such as single or multiple point crossovers.

III. Network Models

We aim to use Genetic Algorithms (GA) to develop an algorithm for finding the minimum cost routes between nodes within a given network. Studies have classified Complex networks into three major models: Random, Small-World and Scale-Free network models [4]–[6]. Each network model is created by following a set design constraints. These constraints are the factors that differentiate between different created network models. Understanding the properties and the structure of the existing network is useful for specifying the strengths and weaknesses of the network and helps toward providing better redesign and utilization techniques, which in term aims toward increasing the performance and minimizing the bottlenecks within the network. In the following sections, we are going to preview three network models: Random, Small-World and Scale-free network models.

A. Random

One of the most basic and popular models of random complex networks is the Erdős and Rényi model introduced in 1959 [4], [7], [8]. In this model, a network with N nodes is constructed along with a probability p for connecting each pair of vertices in the network, excluding duplicate or self looping links. The resulting network represents an Erdős and Rényi random network. Fig. 2 represents an Erdős and Rényi network with 30 nodes and average degree of 3. We will refer to the Erdős and Rényi network model as the Random Network Model.

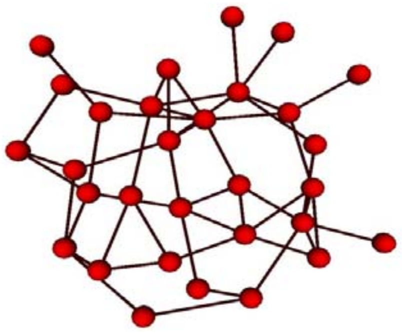
Fig. 2 Erdős and Rényi network with 30 nodes and average degree of 3

B. Small-World

In 1967, Milgram made a famous experiment and found that two random chosen US citizens can reach each other through an average of six social links [5]. In other words, he found that in a social network, everyone in the network could be reached through a finite number of edge transitions [9], [10]. This concept is called the Small-World property, and therefore, the Small-World network model is defined as a network which has the Small-World property. Another property of the Small-World networks is the existence of a large number of loops of size 3 links, i.e. if node *a* is connected to *b* and *c*, then it is highly possible that b and c are also connected to each other forming a cluster of 3 links. A famous random network model which satisfies the Small-World property and has high number of small clusters is the Watts-Strogatz (WS) small-world model [11]. To create a (WS) Small-World network, we start with N nodes, where each node is connected to k neighboring nodes in both directions forming 2.k neighbors. Then, each edge in the network is rewired based on a probability P_r. The resulting network looks similar to the network shown in Fig. 3, for a network of 30 nodes, k=2 and P_r =0.3.

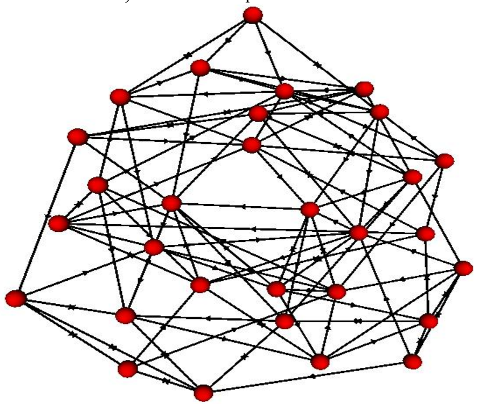
Fig. 3 Small-World network with 30 nodes, k=2 and P_r =0.3

C. Scale-Free

Many studies had shown that the degree distribution for many real networks do not follow a unique pattern, instead they showed that it follows an uneven distribution. Barabasi and Alberts [6] showed that in many systems, there are few nodes in the network that are highly connected while the others have much less connections. These few highly connected nodes represent hubs in the network and they have a large fraction of the existing relations in the network. A network with the mentioned characterization is called a Scale-Free network. Scale-Free networks are built following a growth pattern. First, the network starts with M_0 random nodes with random relations between them. Then the network grows by adding new nodes to the existing network and creating more relations between the newly added node and some of the previous nodes. Adding a new relation between a new node *i* and an existing node *j* is proportional to the total in and out degree of *j*. The probability of creating a link between node *i* and node *j* is represented by the summation of all in and out degree of *j* over the total degree for the network [6] and is given below:

$$p_{i \to j} = \frac{\sum Degree(in, out)_j}{\sum Degree(in, out)_{network}} \qquad (1)$$

Fig. 4 illustrates a scale-free network with 30 nodes, initial population of M_0=9 nodes and connecting probability of P_c=0.1. As shown in Fig. 4, all the nodes are connected in the network and no isolated nodes exist. This is a result of having relatively smaller connecting probability *p*. Also, you can notice that there are four nodes in the middle of the network that almost every other node is connected to, forming hubs for the network. These nodes tend to have highest degree when the network was initialized and therefore, they will have more relations toward them after adding the new nodes to the network.

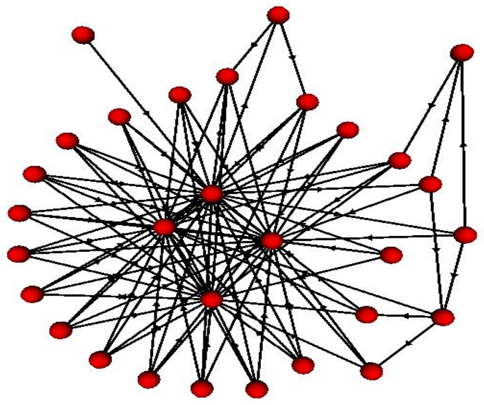

Fig. 4 Scale-Free network with 30 nodes, M_0=30% and P_c = 0.1

IV. Finding Minimum Cost Routes within a Network using GA

In this project, we are going to design and implement an algorithm for finding the minimum cost route between any pair of nodes within a given network. The conducted algorithm is going to be built on the basis of GA. We have selected GA because of its performance and compatibility when dealing with optimization problems. The minimum cost route would have been found using other techniques that can calculate the shortest path between two selected nodes. These algorithms could be used to find all shortest paths between two selected nodes and then evaluating them by calculating the overall cost through each path, in order to specify the minimum cost routes. These techniques sound easier to implement for small size networks with 10 or 20 nodes. However, if the network size is scaled up, these solutions might not be effective anymore because of the large size of the paths available between two nodes within a given network. Therefore, the GA has been adopted due to its adaptive nature where possible solutions are integrated to produce better solutions until reaching the optimal solution. Also, GA could be effectively adopted by any computer-based problem solving system, producing programs that can automatically solve a specified optimization problem with high performance. In the following sections, we are going to describe in more details the implemented GA, which is used to find the minimum cost route between any two nodes within a network.

A. *Algorithm Details*

We have developed a GA based algorithm to solve the problem of finding the minimum cost route between two nodes within a network. The algorithm starts by either reading existing input file or randomly creating the data related to the structure of a network with size N nodes and different inter link weights. Note that if the network is going to be created randomly, you will have the choice to create different types of networks such as Random, Small-World or Scale-Free networks. Based on the selection, the network is going to be generated randomly and according to the specified parameters and constraints. All networks are assumed to be undirected weighted networks, where the weights on the links represent the cost of traversing from a node to another through that specific link. The data related to the network is stored into a two dimensional transition matrix called *network[N][N]*, where *network[i][j]* represents the cost of traveling from node i to node j within the network. After creating the network transition matrix, two nodes are selected to represent a source and destination nodes, and then, the GA is run for a specific number of generations to find the minimum cost route(s) between the selected source and destination. Finally, the above process could be repeated for any pair of source and destination nodes within the existing network in order to get all minimum cost routes between the nodes. Fig. 5 below illustrates the mains steps of the overall algorithm used to find the minimum cost route between the selected source and destinations nodes.

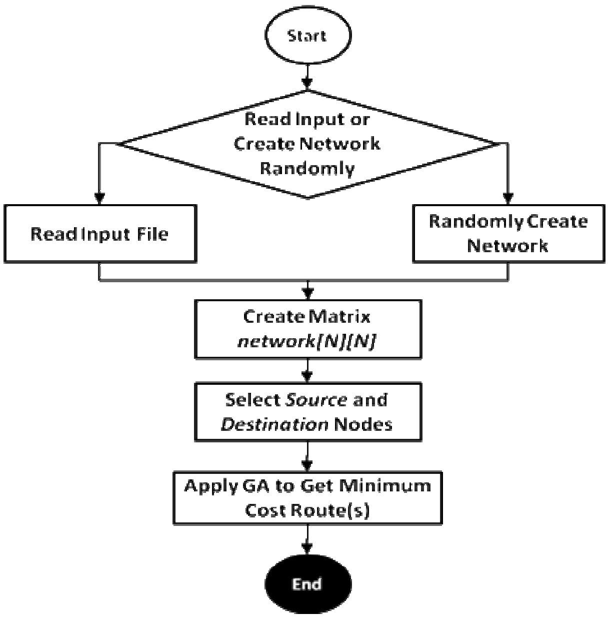
Start
Read Input or Create Network Randomly
Read Input File
Randomly Create Network
Create Matrix network[N][N]
Select Source and Destination Nodes
Apply GA to Get Minimum Cost Route(s)
End

Fig. 5 General overview of the implemented algorithm for finding the minimum cost route between selected source and destination nodes

B. *Implemented GA*

The implemented GA in this project is not different than any other implemented GA in the main steps (generate initial population, selection, crossover and mutation). However, there are some important parameters and constraints that must be specified carefully and accurately in order to produce optimal results using the implemented GA. The related constraints are presented in details in the following sections.

1) *Representation and Initialization parameters:* One of the most important parts of implementing any GA is the representation step. In this step, the designer must carefully choose a form to represent a given solution for the problem. Usually, the solutions are represented as a string of binary bits, where each bit in the solution array represents a gene within a chromosome. Other representation format is the integer representations, where genes are represented by an array of integer values. In this project, we need to calculate

the minimum cost route between any two selected nodes. Therefore, we are going to use integer representation with maximum chromosome size of 2xN, in order to cover variable length solutions. Each solution is represented by an array R[2.N], where each integer R[i] in the array list represents a gene within the chromosome (solution). This gene holds the cost of traversing from one node to the neighboring node on a selected link. For example, if a network with N=10 nodes and s=2 (source node number) and d=6 (destination node number), then the proposed solution R must have 20 genes in maximum, forming an array of integers which represent the cost of traversing from the source to the destination through the selected path. There will possibly be more than one route between any two selected source and destination nodes. Assume you have k routes between two nodes, an example route (solution) could be represented as following:

$R_k[20]$: 20 13 5 48 2 36 0 0 0 0 0 0 0 0 0 0 0 0 0 0

The solution size is selected to be twice the size of the network and therefore, as the size of the network grows, the solution size will also grow, providing larger solution space for routes that can be made with more number of links. This will help in covering most available solutions when generating the initial population as we will present in the next sections.

2) *Fitness Function:* As described in section 2, a fitness function is an objective function that prescribes the optimality of a solution. Fitness function could be described in different forms considering the problem nature and the solution required. In this problem, the minimum cost route between two selected nodes is required to be found and therefore, a meaningful fitness function would be the accumulative summation of the costs along a given path R that connects the two selected nodes, as shown in equation (2). This fitness function is created toward serving the goal of the implemented GA, which is finding the path with the minimum (optimal) cost.

$$Fitness(s,d)_k = \sum_{i=1}^{i=2.N} R_k[i] \qquad (2)$$

3) *Selection and Crossover:* Selection and crossover are two steps used by the GA to generate new population which possibly has better overall fitness. In this project, mutation is not used; instead a single point crossover is used to create the new possible solutions. Single point crossover is done by selecting two parent solutions and making a crossover between them at a single point to generate two new children solutions. At each crossover operation, the crossover point must be randomly selected. To make sure that the generated children are valid solutions, the crossover point must be selected at a common node between the two parents. If the selected crossover point is not common between the two selected parents, then the generated children will not represent valid routes and, therefore will be discarded after the crossover operation. Adding this constraint to the algorithm helps toward saving time by making sure that the generated children solutions represent valid solution after each crossover operation. The selection criteria is based on selecting the routes that have the best fitness (minimum cost) to be stored for the next generation and discard the routes with the worst fitness (maximum cost). This selection criterion is used to hopefully get new generation with better overall fitness, which in term is expected to generate new generations with better overall fitness as well.

V. Discussion and Experimental Results

A C++ code was written to design the algorithm used for finding the minimum cost route between existing nodes within a network. The networks are generated randomly, representing different network models with different sizes. In this report, all networks are undirected networks that are generated randomly with a fixed size of 50 nodes. Also, all links between nodes are assumed to have a unique value α which is referred to as the cost of routing from node *a* to node *b* and vice versa. For this study, we assume that α=1.

In the following sections, we are going to generate different types networks (random, small-world and scale-free) and analyze them by calculating the overall routing cost within the network. The overall routing cost for any network is given as the average of fitness values for all minimum cost routes between existing nodes in the network. To get the average fitness value for a given network, the minimum cost routes are found for every pair of nodes within the network. These minimum values are then averaged to get the overall fitness value for a given network. All experiments are applied for networks with N=50 nodes, with a solution size of *2.N*, and an initial population of size NP=100 solutions. Moreover, the GA is applied on each network for 1000 generations repeatedly until getting the final optimal solution. The designed code for this project will generate the networks automatically after specifying the needed parameters such as the network size, population size, number of generations and etc. Then, the code is going to perform the GA to find the minimum cost routes between every two nodes. Eventually, it is going to get the overall routing cost for the network by averaging all calculated minimum cost routes. For accurate data representation, the procedure is applied for 500 samples of each network and the overall values are averaged to get a more accurate result for each network.

A. Random Network Model: Sample Solution

For random networks with size 50 nodes, a randomly selected network sample is presented in Fig. 6 below. All links are randomly created with no specific pattern and with a creating probability of 0.4 (if the generated random number p<0.4, then create a link between two nodes, otherwise no link is created). Also, all links have a unique value of α, which is removed from the figure for simplicity of the network. As shown in Fig. 6, the network is made of a huge number of randomly created links between network nodes. The final output file for this program contains all paired nodes and the minimum cost paths and their fitness values. A sample part of the final output file is given in Table 1, which represents the minimum cost routes between node number 1 and the first 10 nodes within the network. The maximum and minimum fitness values among all routes are given as 5 and 1 respectively, which means that the longest path between two nodes in the network uses 5 links (5α) and the shortest path between any two existing nodes goes through a single link (1α).

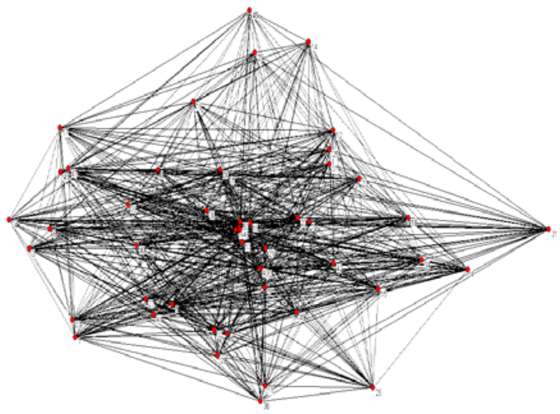

Fig. 6 Random network with N=50 nodes

TABLE I

A SAMPLE OF THE MINIMUM COST ROUTES WITHIN THE GIVEN RANDOM NETWORK AND THEIR FITNESS VALUES

n1	n2	Minimum Fitness	Minimum Cost Route
1	2	2	1-26-2
1	3	2	1-20-3
1	4	2	1-32-4
1	5	2	1-11-5
1	6	1	1-6
1	7	1	1-7
1	8	2	1-11-8
1	9	2	1-19-9
1	10	1	1-10

The total degree (in and out degrees) for the above given random network is 1548 links, which is large but typical for a random network. This is one of the properties of any given random network, since the probability of creating a link between two links is equal to 40 %, which means that there is a 40% chance of creating a link between any two selected nodes upon network creation. On average, the minimum cost routes between selected pairs of nodes have an average fitness value of 1.63 (1.63α), which represents the overall fitness for the network.

B. *Small-World Network Model: Sample Solution*

For the small-world network model, a network of size N=50, rewiring probability P_r=0.5 and number of neighbors k=2 (2.k neighboring nodes) was created and used as shown in Fig. 7 below. As shown in Fig. 7, the resulting networks is presented in a ring like structure in order to better present the small-world properties of the network, which is the connection to the closest 2k neighbors. The effect of the rewiring process during the network creation is clearly presented by the links that are rewired from the closest neighbors to nodes that are located away from the selected node.

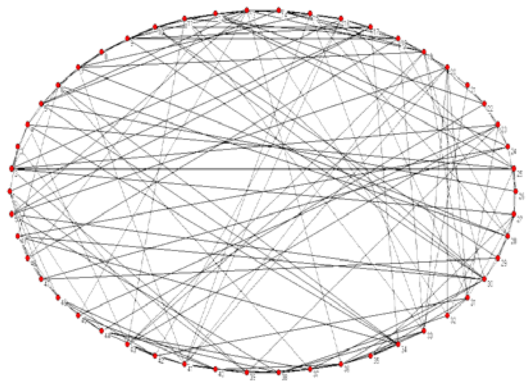

Fig. 7 Small-world network with N=50 nodes, k=2 and P_r=0.5

The final output file for this network is similar in structure to the previously generated output files for the random network model and contains all paired nodes and the minimum cost paths and their fitness values. A sample part of the final output file is given in Table 2, which represents the minimum cost routes between node number 20 and the first 10 nodes with IDs greater than 20. The maximum and minimum fitness values among all routes are given as 9 and 1 respectively, which means that the longest path between two nodes in the network uses 9 links (9α) and the shortest path between any two existing nodes goes through a single link (1α).

TABLE II

A SAMPLE OF THE MINIMUM COST ROUTES WITHIN THE GIVEN SMALL-WORLD NETWORK AND THEIR FITNESS VALUES

n1	n2	Minimum Fitness	Minimum Cost Route
20	21	2	20-33-21
20	22	3	20-33-21-22
20	23	2	20-31-23
20	24	4	20-31-27-26-24
20	25	2	20-32-25
20	26	3	20-31-27-26
20	27	2	20-31-27
20	28	2	20-18-28
20	29	3	20-31-27-29
20	30	3	20-33-36-30

The total degree (in and out degrees) for the small-world network given in Fig. 7 is 346 links, which represents 4 neighbors for each nodes (2k) plus the extra links added after rewiring existing links [(2x2x50) + (rewired links)]. Therefore, the resulted degree for the network is represents a correct value for a small-world network with the given specification. We notice that the total degree for small-world networks is much less than random networks. The average routing cost for the given small-world network, which represents the overall fitness for the network, is found to be 2.45 (2.45 α).

C. Scale-Free Network Model: Sample Solution

A scale-free network with initial population M_0=0.4 and connecting probability P_c=0.06 was created randomly and presented in Fig. 8. The initial population is a portion of the network that is created randomly at the first step. For this network, the first 20 nodes represent the initial random population. As shown in Fig. 8, these nodes reside in the middle of the network. Furthermore, when creating the network, based on the value of the connecting probability P_c and the degree of the nodes within the initial population, a link is created between newly added nodes and some of the existing nodes within the initial population. Nodes 5, 6, 7, 10, 13 and 18 have higher degree among the initial population and therefore, all newly added nodes are connected to them. This is a property of any given scale-free network, which is having hub-like nodes that attract all other nodes toward them.

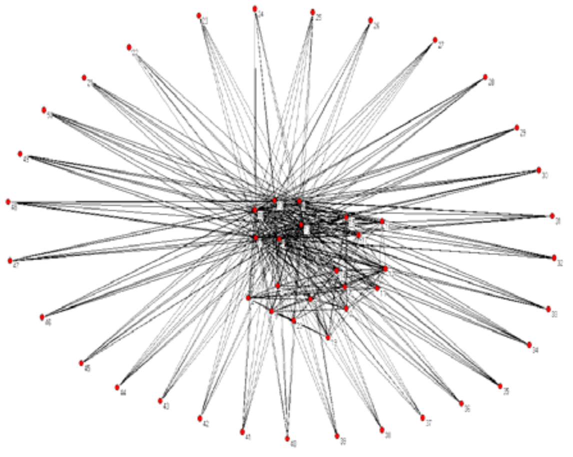
Fig. 8 Scale-Free network with N=50 nodes, M_0=0.4 and P_c=0.06

Similar to the previous cases, a sample part of the final output is given in Table 3, which represents the minimum cost routes between node number 30 and the first 10 nodes with IDs greater than 30.

The total degree (in and out degrees) for the generated scale-free network is 656 links. The total degree depends on the initial population percentage and the number of hub-like nodes within the initial population. On the other hand, the average value of all minimum cost routes which represents the overall fitness value of the network is equal to 2.08 (2.08α), with minimum and maximum fitness values of 1 and 8 links respectively (minimum=1α and maximum=8α).

TABLE III

A SAMPLE OF THE MINIMUM COST ROUTES WITHIN THE GIVEN SCALE-FREE NETWORK AND THEIR FITNESS VALUES

n1	n2	Minimum Fitness	Minimum Cost Route
30	31	2	30-6-31
30	32	2	30-5-32
30	33	2	30-18-33
30	34	2	30-18-34
30	35	2	30-6-35
30	36	2	30-13-36
30	37	2	30-6-37
30	38	2	30-5-38
30	39	2	30-18-39
30	40	2	30-6-40

D. Network Comparison and Concluding Remarks

We have provided sample networks and sample output files in the previous sections. In this section we are going to generate 500 random sample networks of each network model (random, small-world and scale-free) and calculate the resulting average value for the overall fitness for each network type. The results are presented in Table 4 along with the average network degree for the studied network types.

TABLE IV

AVERAGE NETWORK DEGREE AND FITNESS VALUES FOR DIFFERENT NETWORKS AFTER GENERATING 500 SAMPLES

Network Model	Average Network Degree	Average Fitness
Random	1352	1.72
Small-World	359	2.43
Scale-Free	694	2.11

VI. CONCLUSION

As described in this report, GA was used successfully to find the minimum cost path between any two selected nodes within a given network. The used algorithm is consistent, resulting in software with high performance and efficiency. The software finds the optimal solution after relatively small number of generations. The goal of the overall study is to evaluate a given network's structure and propose a redesign criteria (if necessary) based on the calculated minimum cost routes between any two nodes. By calculating the minimum cost routes between any two nodes within the network, the designers can observe and compare the minimum cost routes in terms of number of links and their location, and point out the links that form a bottleneck and have higher traffic rate and suggest redesigning the network in a way to serve the overall routing problem in the network. The bottlenecks might be overcome by simply adding more links, or changing the endpoints of a given link, or by changing the weight on a specific link. These issues were not clear for the designers unless they have an overview of the existing minimum cost routes, which was provided by the suggested GA algorithm in this report.

ACKNOWLEDGMENT

The authors would like to acknowledge the support by Kuwait Foundation for the Advancement of Sciences (KFAS) under a research grant No. 2006-1510-03.

REFERENCES

[1] Genetic Algorithms and Evolutionary Computation By Adam Marczyk, (2004).

[2] Fleming, Peter and R.C. Purshouse. "Evolutionary algorithms in control systems engineering: a survey." *Control Engineering Practice*, vol.10, p.1223-1241 (2002).

[3] Extracted from “http://en.wikipedia.org/wiki/Fitness_function” in April 2010.

[4] P. Erdős and A. Rényi, 1959. On random graphs. *Publicationes Mathematicae*, 6:290–297.

[5] S. Milgran, 1967. The small world problem. *Psychology Today*, 1(1):60–67.

[6] A.-L. Barabási and R. Albert, 1997. Emergence of scaling in random networks. *Science*, 286:509–512.

[7] P. Erdős and A. Rényi, 1960. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci*, 5:17–61.

[8] P. Erdős and A. Rényi, 1961. On the strenght of connectedness of a random graph. *Acta Mathematica Scientia Hungary*, 12:261–267.

[9] D. J. Watts, 1999. *Small worlds : the dynamics of networks between order and randomness*. Princeton University Press.

[10] D. J. Watts, 2003. *Six Degrees. The Science of a Connected Age*. W.W. Norton & Company.

[11] D. J. Watts and S. H. Strogatz, 1998. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442.