

COMPARATIVE EVALUATION OF GENETIC ALGORITHM AND MEMETIC ALGORITHM FOR OPTIMAL TOPOLOGY DESIGN OF COMPUTER NETWORKS

R. O. Oladele¹ & O. T. Oladele, O. T.²

¹*Department of Computer Science, University of Ilorin, Ilorin, Nigeria*

²*Department of Computer Science, Federal University of Technology, Akure, Nigeria*

¹roladele@yahoo.com, ²oladelemrs@yahoo.com

ABSTRACT

Network topology design problem can be formulated as a combinatorial optimization problem. In this paper, a comparative evaluation of genetic algorithm and memetic algorithm is presented with a view to investigating their performance for network design problem. Results from various computation experiments reveal that MA is able to produce better results in term of quality but at a higher computation time. In addition, for a problem instance of small size such as 10-node network design problem GA can produce better solution but at a higher computational time.

Keywords: Memetic Algorithm, Genetic Algorithm, local search, optimization, network

1.0 INTRODUCTION

In NP-hard problems such as topology design problem, local optimization techniques are not effective in searching for a global optimum. For such problems, modern stochastic optimization techniques like Evolutionary Algorithms (EAs) and its variations may be better options. Additionally, some well-known EAs, like Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) are

prone to premature convergence in complex problems and the use of hybridization with other techniques has been shown to improve solution quality. Memetic Algorithm (MA) is a typical hybrid algorithm which is a combination of GA and local search strategy.

In [2] the authors compare the performance of three classic metaheuristic procedures for solving large and realistic versions of network design problem: hillclimbing, simulated

annealing and genetic algorithms. Three alterations that use local search to seed the search or improve solutions during each iteration are also compared. It is shown that employing local search during evolution of the genetic algorithm, a memetic algorithm, yields the best network designs and does so at a reasonable computational cost. Hillclimbing performs well as a quick search for good designs, but cannot identify the most superior designs even when computational effort is equal to the metaheuristics. Garg [9] investigated the relative performance of MA and GA for the cryptanalysis of Simplified Data Encryption Standard (SDES) problem. The algorithms were tested and various experimental results show that MA is slightly superior in accuracy while GA is superior in efficiency. However, for a large amount of cipher text, MA was found to outperform GA. Elbeltagi et al [8] presented a comparison among five algorithms - GA, MA, PSO, Ant Colony Optimization (ACO), and Shuffled Frog-Leaping (SFL). The algorithms were tested with both continuous and discrete optimization problems. The authors found PSO method to be generally better than other algorithms in terms of success rate and solution quality, while being second best in terms of processing time.

This paper investigates the performance differences between GA and MA, in solving a network topology design problem. To the best of our knowledge, papers comparing performance of GA and MA for real-life problem such as network design problem are very rare. Most comparisons in the literature are based on standard test functions and even where real life optimization problem is considered, the optimization function is often single-objective.

1.1 Network Topology Design Problem

Topological design of Wide Area Networks (WANs) involves determining the layout of links between nodes when the mean/peak inter-node traffic is known. The formal definition of network topology design problem is as given below.

1.1.1 Network Design Parameters

For the purpose of this paper, the following network parameters are used.

N denotes the total number of nodes in the network

D_{ij} denotes the physical distance between every pair of nodes i and j

$LFLOW_{ij}$ is the amount of traffic (in Kbps) flowing along link (i, j)

C_{ij} represents the cost of the link between nodes i and j

K_i is the cost of network equipment at node i

CAP_{ij} is the capacity of link (i, j) i.e. the maximum amount of traffic the link can carry at any point in time

P_{ij} is selection status of link (i, j) : $P_{ij} = 1$ if link (i, j) is selected, else $P_{ij} = 0$

R_0 is the minimum reliability required of the network ($R_0 = 0.95$)

L is maximum distance for which the signal is sustained without amplification (L is set to 15km)

A is the cost of each amplifier unit (A is set to #6.00)

Poisson process was used to model the traffic delay

1.1.2 Objective Functions

Two objective functions; network cost and end-to-end message delivery delay

were used [6], each of which is approximated by the following formulation.

i. **Network Cost:**

$$\mathbf{NetCost} = \mathbf{NodeCost} + \mathbf{LinkCost} + \mathbf{AmpCost} \quad (1)$$

Where;

$$\mathbf{NodeCost} = \sum_i \mathbf{K}_i \quad (2)$$

(cost of nodes)

$$\mathbf{LinkCost} = \sum_i \sum_j \mathbf{C}_{ij} \quad (3)$$

(cost of links)

$$\mathbf{AmpCost} = \frac{\sum_i \sum_j \mathbf{D}_{ij} \times \mathbf{A}}{\mathbf{L}} \quad (4)$$

(cost of amplification units on the links)

ii. **Average Delay**

The traffic between nodes is modeled as a Poisson Process. Poisson process has been vastly used in the past to model traffic in conventional networks. For delay function, Poisson traffic model is utilized [1], [6], [7], and is as given in equation (6).

Queueing Delay for Poisson Traffic : The delay in a network for the queueing model is largely due to the queueing of packets in intermediate nodes. The delay formalization can be stated as follows:

$$\mathbf{AvDelay} = \frac{\sum_i \sum_j [\mathbf{DELAY}_{ij} \times \mathbf{LFLOW}_{ij}]}{\sum_i \sum_j \mathbf{LFLOW}_{ij}} \quad (5)$$

From queueing theory, the queueing delay $\mathbf{DELAY}_{ij}$ using standard M/M/1 (Poisson) model is given by:

$$\mathbf{DELAY}_{ij} = \frac{1}{[\mathbf{CAP}_{ij} - \mathbf{LFLOW}_{ij}]} \quad (6)$$

$\mathbf{DELAY}_{ij} = 0$ if there is no link between nodes i and j

$\mathbf{DELAY}_{ij} = \infty$ if the network cannot handle the traffic load with the existing

links' capacities and routing policy.

1.1.3 Constraints

Minimization of Network Cost and Average Delay is done subject to the following constraints:

Flow Constraints which can be expressed as:

$$\mathbf{LFLOW}_{ij} \leq \mathbf{CAP}_{ij} \quad (7)$$

and

Reliability Constraint which can be expressed as:

$$\mathbf{R}(x) \geq \mathbf{R}_0 \quad (8)$$

Monte Carlo Simulation is used to estimate network reliability. The network is simulated t times, given the design and the links' reliabilities. Depth First Search (DFS) is used to implement spanning tree in the algorithm.

1.1.4 Routing Policy

Breadth First Search (BFS) is used for routing. The metric used for this purpose is the length of the link. Given a graph (topology) $G = (V, E)$ and a distinguished source vertex s , BFS computes the distance (smallest number of edges) from s to each reachable vertex.

1.1.5 Assumptions

The following assumptions were made in the problem formulation

- 1 The location of each network node is given
- 2 Nodes are perfectly reliable
- 3 Each $\mathbf{C}_{ij}$ is fixed and known
- 4 Each link is bidirectional i.e. a path can be traversed in either direction
- 5 There is no redundant link in the network
- 6 Links are either operational or failed

1.2 Genetic Algorithm

Genetic algorithms (GAs) are a subclass of evolutionary algorithms introduced by Holland [11] and Goldberg [10] for continuous non-linear optimization, and later extended [5], [4] for combinatorial optimization. GA is a stochastic global search method that simulates the metaphor of natural biological evolution. GA operates on a population of feasible solutions applying the principle of survival of the fittest with the hope of producing better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using crossover operator and mutation operator. This process leads to the evolution of population of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation. This process is repeated until some criterion is met. The general scheme of GA is as shown in Figure 1.

```
BEGIN
    INITIALIZE populations;
    EVALUATE each candidate;
    WHILE (TERMINATION CONDITION
is not satisfied) DO
        SELECT parents;
        RECOMBINE pairs of parents;
        MUTATE the resulting offspring;
        EVALUATE new candidates;
        SELECT individuals for the next
generation;
    END WHILE
END
```

Fig. 1: the general scheme of a GA in pseudocode [13]

1.3 Memetic Algorithm

Memetic Algorithms (MAs) can be seen as “evolutionary algorithms which exploit all available knowledge of the underlying problem” [12] available in the form of greedy heuristic algorithms, local search, specialized recombination operators, or some other ways. MA is motivated by Dawkins’s notion of a *meme* as a unit of information that reproduces itself as people exchange ideas [3]. A key difference exists between genes and memes. Before a meme is passed on, it is typically adapted by the person who transmits it as that person thinks, understands and processes the meme, whereas genes get passed on whole. This thinking has been linked to local refinement, therefore MA is commonly regarded GA that uses local search. Different researches use local search at different points of GA. In this paper, it is used after recombination(crossover) and after mutation. The general scheme of MA is as shown in Figure 2.

```
BEGIN
    INITIALIZE populations;
    EVALUATE each candidate;
    WHILE (TERMINATION CONDITION
is not satisfied) DO
        SELECT parents;
        RECOMBINE pairs of parents;
        LOCAL SEARCH the resulting
offspring;
        MUTATE the resulting offspring;
        LOCAL SEARCH the resulting
offspring;
        EVALUATE new candidates;
        SELECT individuals for the next
generation;
    END WHILE
END
```

Fig 2: The scheme of an MA in pseudocode

1.4 Local Search

The Hill Climbing (HC) Search Algorithm which is a local search technique is used and is shown in Algorithm 3. It is simply a loop that continually moves in direction of increasing quality value [14].

BEGIN

WHILE (TERMINATION CONDITION is not satisfied) **DO**

 newSolution $\leftarrow$ neighbor
 (bestSolution);

IF newSolution IS BETTER THAN
 actualSolution **THEN**

 bestSolution $\leftarrow$ actualSolution;

END IF

END WHILE

END

Fig 3: The general scheme of a Local Search in pseudocode

2.0 Implementation Details

This section provides some useful details regarding the implementation of the above-presented GA and MA.

2.1 Encoding Scheme

The chosen encoding scheme is such that every chromosome codes a possible network, which corresponds to an individual in a set of feasible solutions of the problem. This set of feasible solutions constitutes a population. The chromosome is represented by a constant length integer string representation. The chromosome consists of two parts, the first part contains details of NE's at the nodes and the second part consists of details of the links. For example, if there are H types of nodes, then $\log_2 H$ bits are required to encode a node. Therefore the first part of

the chromosome consists of $N \cdot \log_2 H$ bits, where N is the number of nodes in the network. If a link exists between nodes 1 and 2 then the first bit position in the link part is set to 1. Hence the second part of the chromosome consists of $(N(N-1))/2$ bits.

2.2 Initial Population

The two algorithms start by creating an initial population. There are two ways of generating initial population namely heuristic process and random process. A random process of generating initial populations is adopted. The random initialization procedure does not guarantee the feasibility of each solution in the initial population. As such a checking process is involved. A checking process checks if each solution in the population is feasible (i.e. satisfies the constraints). For all infeasible solutions a repair function is used to replace the infeasible solutions with new feasible solutions.

2.3 Fitness Evaluation

Fitness of a chromosome is evaluated based on principle of Pareto ranking. Pareto-rank of each individual is equal to one more than the number of individuals dominating it. All non-dominated individuals are assigned rank one. Network cost and average delay are used to evaluate the rank of an individual chromosome using the principle of Pareto dominance. The fitness of an individual is given by

$$\text{Fitness} = \frac{1}{\text{Rank}^2} \quad (9)$$

2.4 Selection

Roulette Wheel Selection Process is used. In roulette wheel, individuals are selected with a probability that is directly proportional to their fitness values i.e. an individual's selection

corresponds to a portion of a roulette wheel. The probabilities of selecting a parent can be seen as spinning a roulette wheel with the size of the segment for each parent being proportional to its fitness. Obviously, those with the largest fitness (i.e. largest segment sizes) have more probability of being chosen. The fittest individual occupies the largest segment, whereas the least fit have correspondingly smaller segment within the roulette wheel. The circumference of the roulette wheel is the sum of all fitness values of the individuals. The proportional roulette wheel algorithm procedure is depicted by Algorithm 10. Let $f_1, f_2, \dots, f_n$ be fitness values of individuals 1, 2, ..., n. Then the selection probability, P_i , for an individual i , is given as

$$P_i = \frac{f_i}{\sum_{j=1}^n f_j} \quad (10)$$

The template of the roulette wheel selection procedure is shown below.

```

Procedure: Roulette Wheel Selection
    while population size < pop_size do
        generate pop_size random
number r
        calculate cumulative fitness,
        total fitness ( $P_i$ ) and sum of
        proportional fitness
        (sum)
        Spin the wheel pop_size times
        if sum < r then
            select the first chromosome
        else
            select the jth chromosome
        Endif
    Endwhile
    return chromosomes with
    fitness values proportional to
    the size of the selected wheel
    section
End Procedure

```

2.5 Recombination (Crossover)

This operation operates on two(or more) chromosomes. The chromosomes are randomly selected based on the

probability of crossover which is a randomly generated number ranging between 0 and 10. In this work, two-point crossover technique was implemented. The *crossover probability* (denoted by pC) is the probability of the number of offsprings produced in each generation to the population size (denoted by $popSize$). This probability controls the expected number ($pC \times popSize$) of chromosomes to undergo the crossover operation. A high crossover probability is used here to allow exploration of more of the solution space, and reduces the chances of settling for a false optimum; but if this probability is too high, it results in the wastage of a lot of computation time in exploring unpromising regions of the solution space.

2.6 Mutation

This is the operation of randomly changing some of the bits of the chromosome representing an individual with a view to increasing the exploration of the solution space.

3.0 COMPUTATIONAL RESULTS

This section presents the results obtained when both GA and MA were applied to solve three instances of topology design problem. Comparative analysis of GA and MA results are as well discussed. All experiments were performed on a HP 630 NOTEBOOK PC with the following configuration: 2.13GHz Processor Speed, 3.0GB RAM and 64 BIT OS

The two algorithms were implemented in Java. For each algorithm 10 runs were made for each test problem. The sample tables of result below report the non-dominated objective vectors and the CPU time used to obtain the results. The

results in bold (objective vectors) are the best pareto-ranked vectors for each run (that is, part of the pareto optimal front). The algorithms were run with the following parameters:

Population size - 100 (250 for test problem 3)

Mutation probability – 0.02

Two- point crossover was used

Number of Node Type – 4

The test problems are:

Problem 1: 10-node network design

Problem 2: 21-node network design

Problem 3: 36-node network design.

Tables of Results for Problem 1 (GA versus MA)

Table 1: GA and MA Results of Run 1 for Problem 1

GENETIC ALGORITHM					MEMETIC ALGORITHM				
No of Gen	C/Ratio	Cost	AvDelay	CPU Time	No of Gen	C/Ratio	Cost	AvDelay	CPU Time
5	98	645.4	0.07	126	5	98	619.6	0.06	145
10	100	645.4	0.06	345	10	100	619.6	0.05	410
15	100	645.4	0.06	459	15	100	619.6	0.05	548
20	100	645.4	0.06	627	20	100	619.6	0.05	643

Table 2: GA and MA Results of Run 2 for Problem 1

GENETIC ALGORITHM					MEMETIC ALGORITHM				
No of Gen	C/Ratio	Cost	AvDelay	CPU Time	No of Gen	C/Ratio	Cost	AvDelay	CPU Time
5	100	777.8	0.07	154	5	98	534.0	0.06	166
10	100	777.8	0.06	384	10	100	534.0	0.05	472
15	100	777.8	0.06	459	15	100	534.0	0.06	537
20	100	534.0	0.03	572	20	100	534.0	0.07	615

Tables of Results for Problem 2 (GA versus MA)

Table 3: GA and MA Results of Run 1 for Problem 2

GENETIC ALGORITHM					MEMETIC ALGORITHM				
No of Gen	C/Ratio	Cost	AvDelay	CPU Time	No of Gen	C/Ratio	Cost	AvDelay	CPU Time
5	98	1305	0.07	837	5	97	1259	0.06	954
10	100	1167.4	0.06	1482	10	100	1167.4	0.05	1892
15	100	1265.8	0.05	4143	15	100	1167.4	0.04	4467
20	100	1265.8	0.06	5434	20	100	1158.6	0.04	5665

Tables of Results for Problem 3 (GA versus MA)

Table 4: GA and MA Results of Run 1 for Problem 3

GENETIC ALGORITHM					MEMETIC ALGORITHM				
No of Gen	C/Ratio	Cost	AvDelay	CPU Time	No of Gen	C/Ratio	Cost	AvDelay	CPU Time
5	98	1305	0.05	2333	5	97	1167.4	0.06	3598
10	100	1259	0.07	3771	10	100	1167.4	0.05	5890
15	100	1259	0.08	7043	15	100	1167.4	0.05	9673
20	100	1167.4	0.06	12980	20	100	1167.4	0.05	15897
25	100	1259	0.09	13090	25	100	1167.4	0.04	18754

Table 5: GA and MA Results of Run 2 for Problem 3

GENETIC ALGORITHM					MEMETIC ALGORITHM				
No of Gen	C/Ratio	Cost	AvDelay	CPU Time	No of Gen	C/Ratio	Cost	AvDelay	CPU Time
5	98	1305	0.07	2454	5	97	1259	0.06	3987
10	100	1167.4	0.06	3672	10	100	1167.4	0.05	5673
15	100	1259	0.06	7112	15	100	1167.4	0.05	9210
20	98	1167.4	0.05	9716	20	100	1167.4	0.05	14900
25	100	1167.4	0.05	12273	25	100	1167.4	0.05	18634

4.0 DISCUSSION AND ANALYSIS OF RESULTS

For 10-node network problem, the results of 10 experimental runs of both algorithms follow two different trends. The first trend (hereafter called Trend A) is depicted by Table 1 whereby the quality of MA results and the computational cost (in time) are better and higher than those of GA respectively and this happened to be the case for most of the runs. The second trend (hereafter called Trend B) is depicted by Table 2 whereby GA produced results which are slightly better than those of MA results but at a higher computational cost.

In the case of 21-node network problem, the results 10 experimental runs of both algorithms follow just one trend – Trend A. This trend is depicted by Table 3 whereby the quality of results and the computational cost of results for MA are better and higher respectively than those of GA.

For 36-node network problem, the results of 10 experimental runs of both algorithms follow two different trends. The first trend which happened to be trend A is depicted by Table 4, and this is the trend of results obtained for most of the runs. The other trend (hereafter called Trend C) is depicted by Table 5 whereby MA and GA rank the same in term of results quality and MA is superior to GA in term of computation time.

Generally speaking, it appears (from the results) that the difference between quality of MA results and the quality of GA results increases as the network size increases. The fact that GA is very susceptible to premature convergence particularly for complex optimization

problems such as network design problem is supported by the low quality results obtained from GA for most of the runs for the 3 problem instances.

6. CONCLUSION

The relative performance of GA and MA were investigated for optimal topology design of computer network. The two algorithms were tested with randomly generated networks of 10-node size, 21-node size and 36-node size. Overall, MA outperformed GA in terms of solution quality while GA outperformed MA in terms of computation time (CPU time). For 10-node network however, GA has the tendency to produce results that are of better quality compared to MA result but this this will be with a computation time higher than that of MA. The major contribution of this paper is: extensive investigation of performance differences between GA and MA for multi-objective optimization of network design. The optimization model used in this work is limited to two objectives and two constraints. It is also pertinent to point out that the optimization model presented in this paper was only tested for randomly generated network data. For future work, the model will be expanded to include more objectives and more constraints. Testing such expanded model using real network data will also be of interest.

ACKNOWLEDGMENTS

Many thanks to Hammed Sanusi for programming assistance.

REFERENCES

- [1] K. K. Aggarwal, Y. C. Chopra, J. S. Bajwa. Reliability evaluation by network decomposition. *IEEE Trans Reliab*, R-31:355–358, 1982.
- [2] F. Altıparmak, B. Dengiz, A. E. Smith. Optimal design of reliable computer networks: A comparison of metaheuristics. *Journal of Heuristics*, 9 (6):471-487(17), 2003.
- [3] N. B'elanger, G. Desaulniers, F. Soumis, J. Desrosiers. Periodic airline fleet assignment with time windows, spacing constraints, and time dependent revenues. *European Journal of Operation Research*, 175(3):1754-1766, 2006
- [4] J. E. Biegel, J. J. Davern. Genetic algorithms and job shop scheduling. *Computers and Industrial Engineering*, 19(1-4):81-91, 1990.
- [5] J. Cohoon, S. U. Hedge, W. N. Martin, D. S. Richards. Distributed genetic algorithms for the floor plan design problem. *IEEE Transactions on Computer Aided Design*, 10(4): 483-492, 1991.
- [6] K. A. De Jong. An analysis of the behaviour of a class of genetic adaptive systems. *PhD thesis*, University of Michigan, Ann Arbor, MI, USA, 1975.
- [7] A. Dutta, S. Mitra. Integrating heuristic knowledge and optimization models for communication network design. *IEEE Transactions on Knowledge and Data Engineering*, 5:999 – 1017, 1993.
- [8] E. Elbeltagia, T. Hegazy, D. Grierson. Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics*, 19 : 43 – 53, Elsevier 2005.
- [9] P. Garg. A Comparison between Memetic algorithm and Genetic algorithm for the cryptanalysis of Simplified Data Encryption Standard algorithm. *International Journal of Network Security and Its Applications (IJNSA)*, 1(1) : 34 – 42, 2009.
- [10] D. E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.
- [11] J. Holland. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, 1975.
- [12] P. Moscato. Memetic algorithms: A short introduction.
- [13] Yew-Soon Ong, Meng-Hiot Lim, Ning Zhu, and Kok-Wai Wong. Classification of adaptive memetic algorithms: A comparative study. *IEEE Transactions on Systems, Man, and Cybernetics*, 36(1), 2006.
- [14] S. J. Russell, P. Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall, 2003.

TEST DATA

10-NODE NETWORK

Node Details (NodeType, K_i) = {
(01,42) , (01, 78) , (10,33) , (00,53)
, (01,42) , (00,13) , (10,9) , (11,23)
, (10,57) , (10,25) }

Link Details (D_{ij} , C_{ij} , CAP_{ij} ,
 $LFLOW_{ij}$) = { (28,47,60,46)
, (20,43,90,72) , (28,12,54,28)
, (62,39,61,46) , (42,23,24,9)
, (42,30,16,14), (36,3,44,16)
, (40,18,75,54) , (10,36,29,8)
, (44,30,79,53) , (44,45,54,35)
, (36,18,66,51) , (32,13,78,25)
, (14,16,96,54) , (16,13, 84,74)
, (21,28,76,17) , (22,3,80,71)
, (3,39,55,54) , (47,12,66,62)
, (26,11,89,56) , (13,42,77,47)
, (46,22,45,39) , (28,6,53,16)
, (5,38,89,57) , (28,40,16, 9)
, (48,49,49,40) , (18,34,37,9)
, (34,35,11,8), (11,41,39,31)
, (46,20,32,9) , (11,3,50,35) , (70,1,54,41)
, (18,6,8,65) , (35,42,91,66)
, (14,33,10,26) , (11,33,60,9)
, (43,16,79,49) , (20,43,88,56)
, (16,13,96,68) , (6,30,91,67)
, (34,49,16,7) , (37,21,57,49)
, (20,12,79,62) , (33,46,81,70)
, (48,25,8,7) }

21-NODE NETWORK

Node Details = { (01,42) , (01,78)
, (10,33) , (00,53) , (01,42) , (00,13)
, (10,9) , (11,23) , (10,57) , (10,25)
, (01,53) , (00,55) , (01,11) , (00,34)
, (10,33) , (00,32) , (01,51) , (10,38) ,
(10,15) , (10,57) }

Link Details = { (29 20 87 74) , (4 15 75
43) , (13 47 50 35) , (41 16 69 52) , (32
25 72 54) , (43 42 89 63) , (31 1 75 21) ,
(29 38 76 70) , (1 33 39 11) , (16 37 59

56) , (30 44 70 42) , (6 47 89 77) , (10 12
81 18) , (30 6 66 37) , (26 3 80 54) , (17
45 98 33) , (12 10 49 39) , (32 9 61 31) , (6
45 35 24) , (14 42 67 24) , (19 10 89 31)
, (6 31 73 28) , (31 19 29 22) , (12 21 80 3)
, (49 35 60 13) , (15 30 95 39) , (15 3 29 9)
, (4 2 82 64) , (26 27 27 12) , (24 42 57 2)
, (25 46 68 66) , (23 18 47 14) , (5 28 94
65) , (30 18 44 26) , (7 18 78 59) , (20 44
80 38) , (33 29 30 7) , (18 10 99 5) , (25 43
18 2) , (30 30 94 13) , (0 26 82 46) , (22 0
87 52) , (40 6 63 15) , (10 41 46 3) , (25 45
35 10) , (15 22 35 56) , (46 28 32 5) , (13 8
31 47) , (17 18 29 35) , (5 24 89 70) , (36
25 90 76) , (32 20 94 75) , (40 34 84 73)
, (7 29 94 53) , (39 35 33 21) , (37 42 57 4)
, (43 41 71 60) , (20 28 85 45) , (36 17 51
19) , (22 19 83 48) , (44 17 28 19) , (36 37
40 19) , (32 36 40 14) , (4 12 88 78) , (32
47 88 8) , (48 19 27 7) , (26 7 73 60) , (28
13 32 1) , (20 21 47 19) , (41 28 84 54)
, (30 28 78 66) , (20 38 92 67) , (21 27 88
27) , (37 21 63 56) , (27 22 57 35) , (3 48
39 38) , (20 32 62 56) , (17 33 74 60) , (41
24 60 14) , (11 4 93 44) , (20 44 75 74)
, (49 30 73 52) , (39 16 64 57) , (12 40 62
54) , (33 16 12 70) , (43 20 83 48) , (0 16
93 71) , (23 29 40 8) , (2 35 81 36) , (11 38
78 62) , (7 11 93 63) , (0 33 94 74) , (9 48
88 54) , (9 46 86 69) , (15 44 87 32) , (12
18 51 43) , (16 24 79 43) , (28 8 68 7) , (41
49 67 27) , (29 24 78 60) , (48 5 63 12)
, (18 22 23 1) , (22 31 17 7) , (14 45 58 4)
, (3 45 73 64) , (16 28 89 71) , (5 8 59 26)
, (16 49 65 50) , (6 25 39 9) , (48 35 76 73)
, (1 30 35 63) , (19 29 82 34) , (35 27 57
20) , (43 10 73 70) , (17 28 25 15) , (44 30
14 3) , (5 20 63 12) , (19 40 46 59) , (8 30
82 50) , (7 9 5 54) , (7 10 74 66) , (30 14
18 55) , (7 5 83 15) , (30 33 69 64) , (15 10
64 61) , (33 11 14 2) , (18 31 79 75) , (2 38
66 0) , (47 0 29 17) , (20 29 52 48) , (46 38
10 4) , (5 45 75 40) , (39 17 83 66) , (18 3
94 4) , (30 25 60 43) , (31 32 84 71) , (34
45 74 74) , (5 19 68 42) , (27 48 72 69)
, (13 6 45 33) , (20 17 37 23) , (41 26 97
30) , (34 42 54 22) , (5 42 63 47) , (39 26

**Comparative Evaluation Of Genetic Algorithm and Memetic Algorithm for Optimal Topology
Design of Computer Networks
R.O. Oladele & O. T. Oladele**

71 47) ,(18 28 20 15) ,(18 4 70 10) ,(16 40) ,(5 2 97 78) ,(8 13 67 57) ,(42 21 68
12 87 32) ,(1 13 97 11) ,(27 39 71 62) 47) ,(5 20 84 23) ,(41 28 61 51) ,(41 38
,(41 14 94 59) ,(1 33 63 57) ,(12 2 70 43) 70 31) ,(8 48 80 59) ,(9 10 97 25) ,(40 38
,(37 4 77 51) ,(25 16 85 21) ,(8 17 40 25) 86 55) ,(31 20 65 60) ,(39 39 41 40) ,(22
,(6 11 87 42) ,(48 47 97 21) ,(28 39 42 40 95 77) ,(44 16 72 56) ,(21 45 88 53)
21) ,(18 21 19 15) ,(46 12 99 51) ,(1 27 29 2 74 37) ,(21 45 60 40) ,(4 37 37 7)
99 76) ,(8 31 21 11) ,(6 13 91 36) ,(27 17 12 43 48 38) ,(3 43 63 48) ,(46 25 63
69 21) ,(18 16 80 38) ,(42 20 92 19) ,(38 43) ,(4 31 53 51) ,(18 9 70 22) ,(47 9 76
33 66 33) ,(47 5 57 27) ,(39 3 45 19) ,(30 70) ,(38 13 75 44) ,(20 49 83 52) ,(21 15
4 88 78) ,(39 14 60 30) ,(28 40 92 41) ,(2 90 25) ,(9 45 66 49) ,(42 40 55 0) ,(1 37
48 63 22) ,(17 10 83 42) ,(30 17 71 50) 80 27) ,(40 8 88 28) ,(9 27 73 38) }

,(14 20 66 79) ,(10 27 83 65) ,(43 27 89