ISSN 2047-3338

Comparative Analysis of Genetic and Simulated Annealing Algorithms for Road Traffic Congestion Management

AKINOLA Solomon Olalekan[1] and ABDULHAMEED Idris A.[2]
[1,2]Department of Computer Science, University of Ibadan, Ibadan, Nigeria

***Abstract*– Road congestion is an optimization problem because of the difficulty it poses and its high unpredictable nature. Simulated Annealing (SA) and Genetic Algorithm (GA) have been applied to optimization problems; however, trade-offs exist as to which of the algorithms converges faster and the cost attached. Searching the optimal solution to the problem domain within a reasonable period is also a problem. This study is focused on performance evaluation of GA and SA as applicable to a crossroad and an adjoining T-junction Traffic congestion problem. The algorithms were simulated and implemented with MATLAB. The fitness values and traffic decongestion times for 10 cycles were measured. Statistical T-test was then used to compare their performances. The result showed that there was higher decongestion time in SA compared to GA with the peak values of 51 and 48 seconds respectively. The statistical test showed that, even though there were mean differences between the two techniques, there were no statistical significant differences between their performances.**

***Index Terms*– Simulated Annealing, Genetic Algorithm and Road Traffic Congestion**

I. INTRODUCTION

IN 1982, the method of Simulated Annealing (SA) was introduced by Kirkpatrick *et al*., [1]. Initially, it was very popular for Integrated Circuit (IC) chip design but now it has a wide range of applications -timetable problem, graph partitioning and transportation systems. Simulated Annealing (SA) is one of the divisions of Artificial intelligence (AI) in Computer Science. Generally, SA is a probabilistic method for finding the global minimum of a cost function that may have several local minima [2]. According to Jeff [3], Simulated Annealing is defined as a programming method which tries to simulate the procedure of annealing – a physical process for cooling solids slowly. This occurs at a minimum energy configuration.

One of the advantages of SA is that it converges to the optimum solution of any optimization problem. It also works as a hill-climbing search method that allows moves in less good goal directions once in a while to escape local minima. However, the algorithm is usually not faster than its contemporaries [4]. Talking about the applications of SA, Johnson *et al.* [5], [6], [7], as cited in Dimitris and John [2], discuss the performance of SA on four problematic areas: travelling salesman problem (TSP), graph partitioning problem (GPP), graph colouring problem (GCP) as well as number partitioning problem (NPP). The researchers (i.e., Johnson et al) apply SA to these NP-hard problems by means of a cooling schedule in which the temperature decreases in a geometric progression, namely, $T(t+1) = rT(t)$.

On the other hand, Genetic Algorithm (GA) was invented by Goldberg with the motivation from Darwin's theory of evolution. Genetic algorithm was first developed in the mid-1960s [8] and it is an area of Artificial Intelligence (AI). The possible solutions to a problem are referred to as chromosomes and a diverse set of chromosomes is grouped into a gene pool. The worth of these solutions is determined using a fitness function, and this quality is used to determine whether or not the chromosomes will be used in producing the next generation of chromosomes. The contents of high quality solutions are more likely to continue into the next generation. The next generation is generally formed via the processes of crossover, combining elements of two chromosomes from the gene pool, and mutation, by randomly altering elements of a chromosome. In terms of application, Genetic algorithms are widely used for finding good excellent solutions for a diverse variety of difficult problems in science, engineering and mathematics [9], [10] as cited in Steven and Brian, [11].

In other words, GA is an adaptive stochastic search algorithm. Stochastic searches are those that make use of probabilities to help achieve their searches. They are usually referred to as function optimizers in the literature [12]. GA is a family of models used in computation. These algorithms encode a possible solution to a given problem on a simple chromosome-like data structure and subsequently apply recombination operators to these structures to preserve critical information. The areas of application of the GA are – image processing, pattern recognition, machine learning, road traffic management and so on [13]. Furthermore, the possibility to use genetic algorithms to solve real world problem has been tested and evaluated by the researchers. Gustaf [14] applied the concept to road traffic management. This researcher considered the standard genetic algorithm. Genetic algorithms are stochastic and the goal is to achieve best results. They typically consume many resources in terms of CPU time and memory size.

II. SYSTEM OVERVIEW

A) The Interface Development

The interface was developed with MATLAB. It consisted of a GUI (Graphical user Interface) with a collection of tools – pushbutton, check boxes; drop-down buttons, edit buttons, axes and option boxes as shown in Fig. 1. The model functioned after following these procedures. The user first checked the type of algorithm to use (SA or GA), then the number of cycle was entered in the box. Then, the user clicks the calculate button to begin the program.

B) Design Constraints

One of the design constraints in the design is the signal indicators. The signal indicators consisted of 'red' and 'green' lights. Red and Green light indicators were used for the implementation because of the trade-off between signal and lines of codes. The assumptions made during the system design were:

a) That the lengths or widths of lanes were equal
b) That traffic decongestion was dependent on time
c) That the lanes were doubled
d) That the road users had equal priorities
e) The traffic queue involved generation of pseudorandom numbers to substitute for the number of vehicles in the real-world scenario. The intersection consisted of a cross-road and an adjoining T-junction.
f) That there was no bulb or other device failure
g) That the system was interface driven

C) System Architecture

System Hardware Architecture

The traffic flows depicted in Fig. 1 were implemented on Microsoft Windows 7 Professional (Version 6.1.7601 Service Pack 1 Build 7601); System Model: HP 630 Notebook PC; System Type: X86-based PC; processor: Intel(R) Pentium(R) CPU B960 @ 2.20GHz, 2200 MHz, 2 Core(s), 2 Logical Processor(s) and Main memory of 2048MB.

System Software Architecture

Standard Simulated Annealing Flowchart: The original SA flowchart as presented by Luke [15] was modified to solve the problem of traffic congestion in this study. It begins with the initial solution to the problem given, which is also the *Best* solution so far, and the temperature set at the initial temperature Temp (i).This solution becomes the *Current* solution and the *Parent* or active solution. The number of Monte Carlo (ITRY) simulation attempts was set to 0.

ITRY is incremented by 1 and is tested to see if it has reached the maximum number of attempts at this temperature. If so, the current temperature is checked. If it is equal to the final temperature, Temp (f), the simulation halts and both the final and the *Best* solutions found during the simulation are obtained. If the current temperature is above the final temperature, it is reduced using a cooling schedule. The number of Monte Carlo attempts (ITRY) is reset to 1.

Fig. 1: The Developed model's Interface

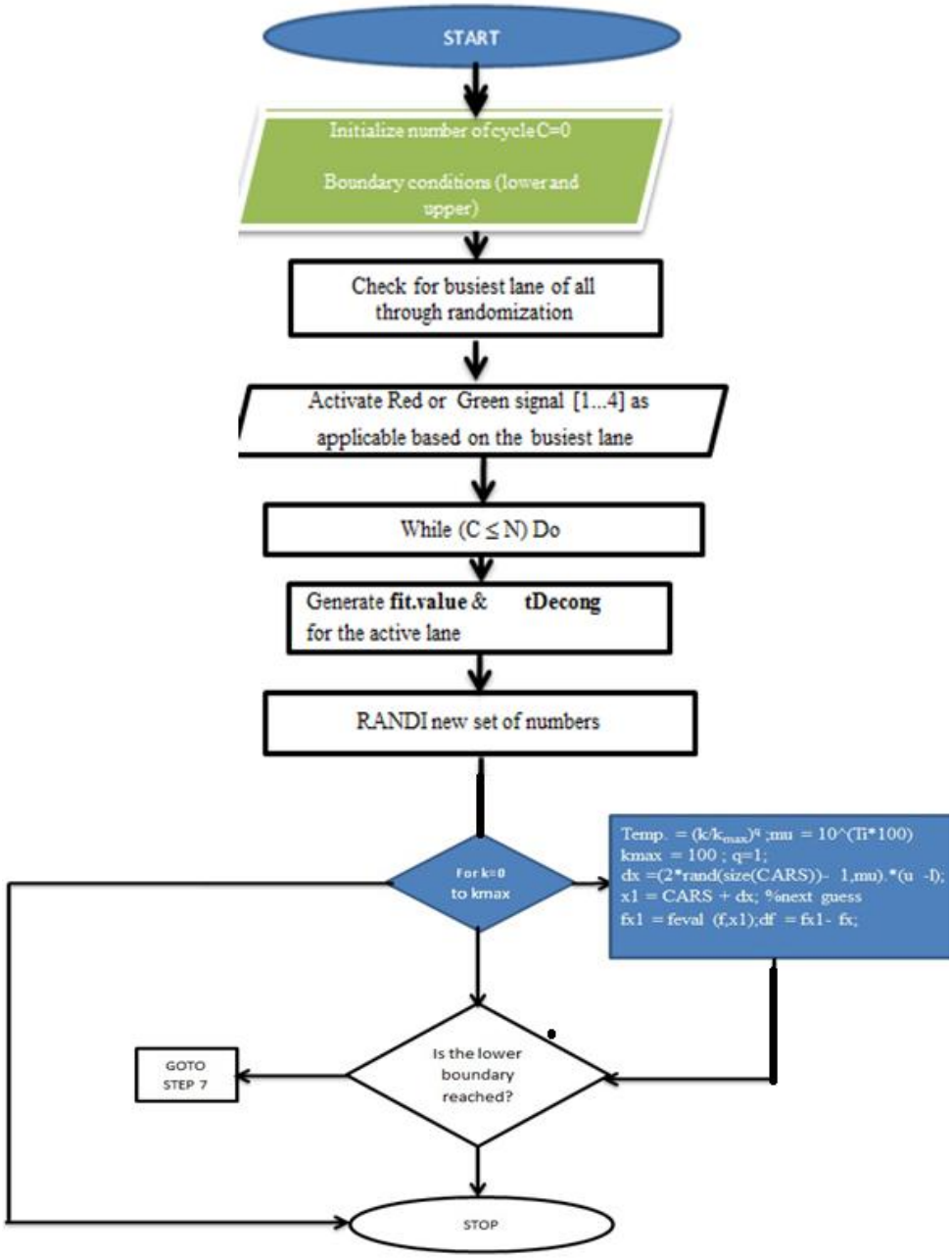

Fig. 2: The Modified SA Flowchart

If the number of attempts at this temperature has not been reached, or the temperature has been decreased, the *Parent* solution is modified to generate a *New* solution. If the energy of the *New* solution is lower than that of the *Parent*, it is checked to see if it is the *Best* solution found to date. If it is, it is stored separately. Whether or not it is the *Best*, it becomes the new *Parent* solution for the next Monte Carlo step. Whenever the *Parent* solution is updated, so is the *Current* solution.

If the energy of the *New* solution is higher than the *Parent's* by an amount dE, the Boltzmann probability ($e^{-dE/kT}$ where k is Boltzmann's constant and T is the current temperature) is calculated. If this probability is greater than a random number (Ran) between 0.0 and 1.0 this *New* solution is accepted and becomes the *Parent* solution for the next iteration, and the *Current* solution. On the other hand, if the Boltzmann probability is less than Ran, the *New* solution is rejected and the *Current*/*Parent* solution stays the same and is used in the next iteration.

ALGORITHM 1: The Standard SA Algorithm (Mohammad, 2008)

```
1.  s = Generate_Initial_Solution()
2.   T = T_0
3.   WHILE termination conditions not met
4.       s1 = Pick_At_Random(N(s))
5.       IF f(s1) < f(s)
6.              s = s1
```

7. ELSE
8. Accept s1 as new solution with probability p(T,s1,s)
9. ENDIF
10. Update(T)
11. ENDWHILE

ALGORITHM 2: Modified SA Algorithm

1. Initialization: The number of cycles C = 0
2. Randomization: check for the busiest lane using pseudorandom function;
3. Check for the busiest of the lanes;
4. Activate RED or GREEN signal as applicable to the busiest lane
5. While C<=N do
6. Generate fitness value and decongestion time (tDecong) for the active lane;
7. Repeat
8. Randomize new set of numbers;
9. For k = 0 to kmax do
10. Temperature = $(\frac{k}{kmax})^q$
11. mu = 10^(T_i*100); kmax = 100; q=1; fx1=feval (f,x1); df = fx1-fx;
12. until lower boundary is reached;
13. stop;

The Standard Genetic Algorithm: Kenneth [16] presented the flow diagram of the standard Genetic Algorithm procedure. The standard flow diagram was modified to solve the problem of road traffic congestion. The modified flowchart is depicted in Fig. 3 Genetic algorithm implements the genetic algorithm at the command line to minimize an objective function.

The genetic algorithm uses three main types of rules at each step to create the next generation from the current population:

(i) *Selection rules* select the individuals, called *parents*, which contribute to the population at the next generation.
(ii) *Crossover rules* combine two parents to form children for the next generation.
(iii) *Mutation rules* apply random changes to individual parents to form children.

In this study, the following functions and variables were used:

(i) Function chrms2 = crossover(chrms2, Nb):This created crossover between two chromosomes
(ii) Stopping criteria: This was set at maxit=100;
(iii) Popsize: The set population size was 20
(iv) Mutate: Represented with 0.1
(v) The selection was set at 0.5

ALGORITHM 3: The Modified GA

1. Start
2. Generate initial population [1..20] in the current population
3. Compute fitness for individuals 1..n in the current population
4. Specify selection probability (sProb) = 0.5; mutation rate (mrate) = 0.25, and crossover rate (cRate) = 0.75
5. While fit_value of chromosome >= sProb do
6. Perform crossover
7. Perform mutation
8. New chromosome (offspring)
9. Return result
10. Stop

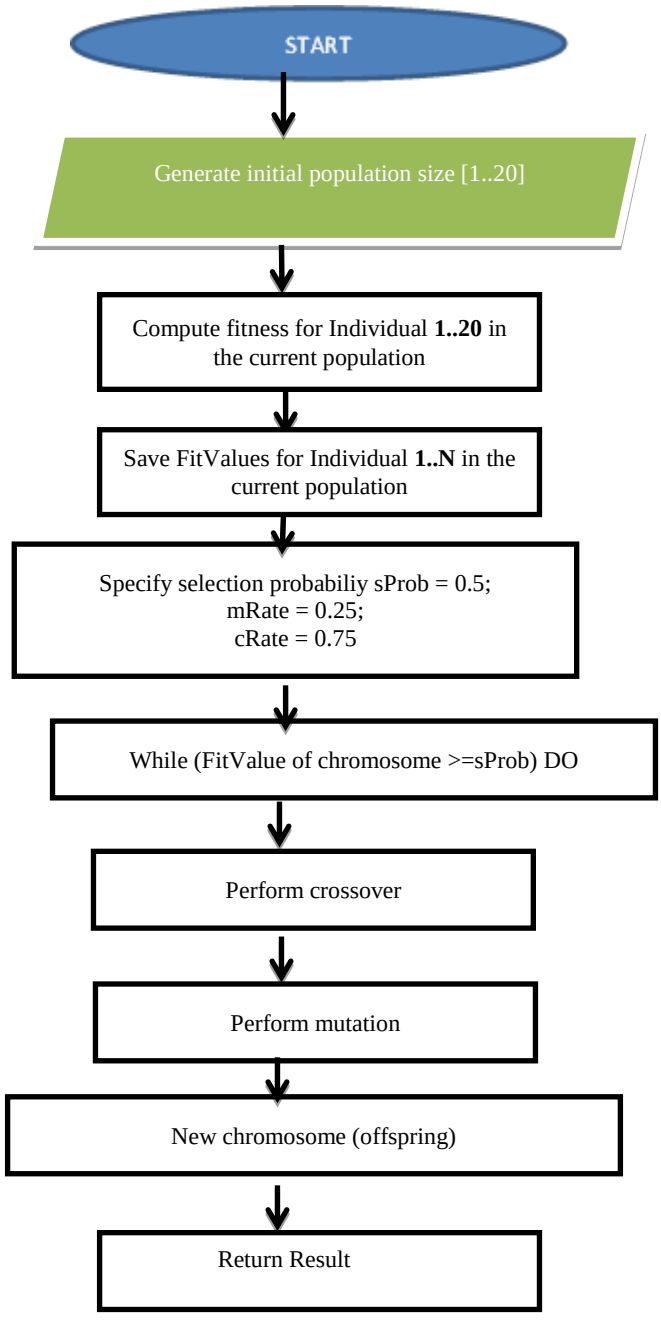

Fig. 3: The modified GA Flow diagram

III. RESULTS AND DISCUSSION

A) Results for Simulated Annealing

From Fig. 4, when a user selects the algorithm to use, for instance, SA, the arrow shows the direction of movement of vehicles generated randomly (i.e., North, South, East and West). The simulation result reported the system time for every instance (during same cycle). From the same Figure, the number of cycle entered was 10. At the end of the cycle the decongestion time was recorded alongside the fitness function. The maximum number of routes or directions for distribution of vehicles was five.

However, Table I shows that cycle four had the highest decongestion time, while equal time occurred at cycles nine and ten. From the second column of Table I, it was observed that the minimum and maximum fitness functions occurred at cycles seven and nine respectively. It was deduced that, cycle four (4) took the highest decongestion time of 51s.

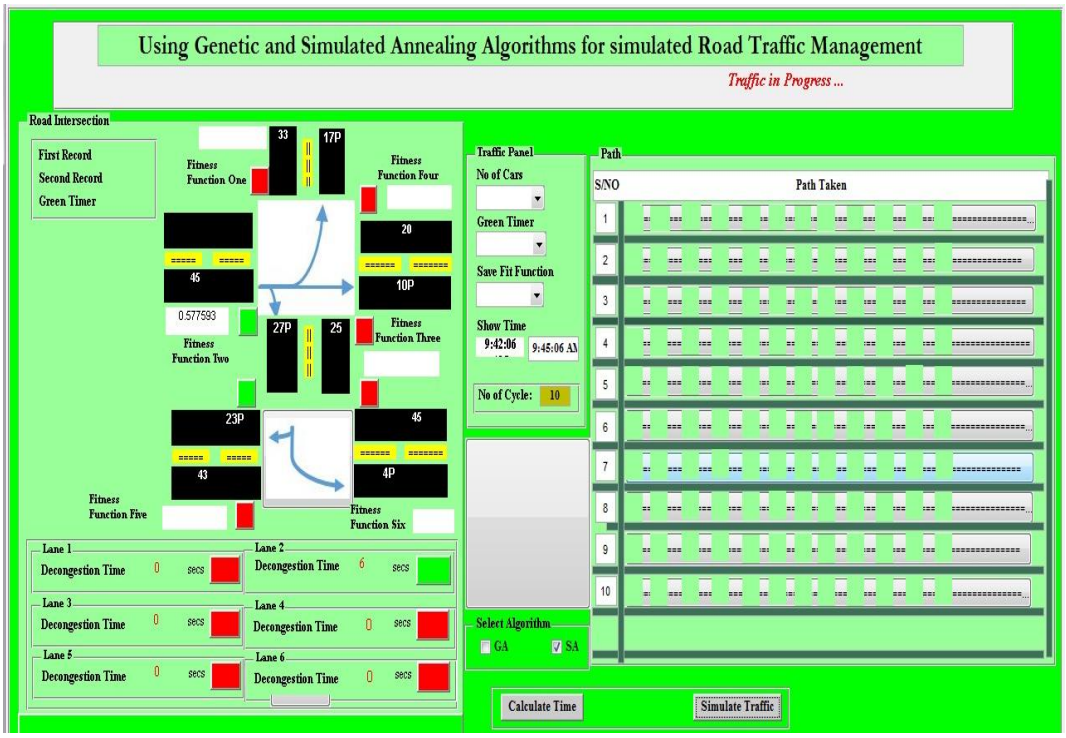

Fig. 4: Traffic simulation at C= 10 (Simulated Annealing)

Table I: Fitness Value vs. Time to Decongest in SA

Cycle	Decongestion Time t_d (s)	Fitness function
1	11.0	0.04103
2	9.0	0.02167
3	30.0	0.34140
4	51.0	0.08911
5	15.0	0.23800
6	50.0	0.47010
7	40.0	0.00002
8	11.0	0.16500
9	6.0	0.78180
10	6.0	0.57760

B) Results for Genetic Algorithm

Just like in Fig. 4, the user selects the algorithm to use (GA) from the interface. The arrows show the direction of movement of vehicles generated randomly (i.e., North, South, East and West). The simulation result recorded the system time for every instance (during same cycle). From the same figure, the number of cycle entered was 10. At the end of the cycle the decongestion time was recorded alongside the fitness function (Table II). The maximum number of distributed routes was equally five as in the first case. It was observed that there were closeness in values of decongestion time at Cycles 1, 4, 8 and 10. There existed, equal fitness values at Cycles 8 and 9 as shown in Table II. In addition, the fitness values generated were all between the range – 0.2000 and 0.2357.

Table II: Fitness Value vs. Time to Decongest in GA

Cycle	Decongestion Time t_d (s)	Fitness function
1	48.0	0.2335
2	18.0	0.2337
3	11.0	0.2336
4	48.0	0.2341
5	18.0	0.2401
6	27.0	0.2341
7	14.0	0.2357
8	47.0	0.2340
9	35.0	0.2340
10	47.0	0.2349

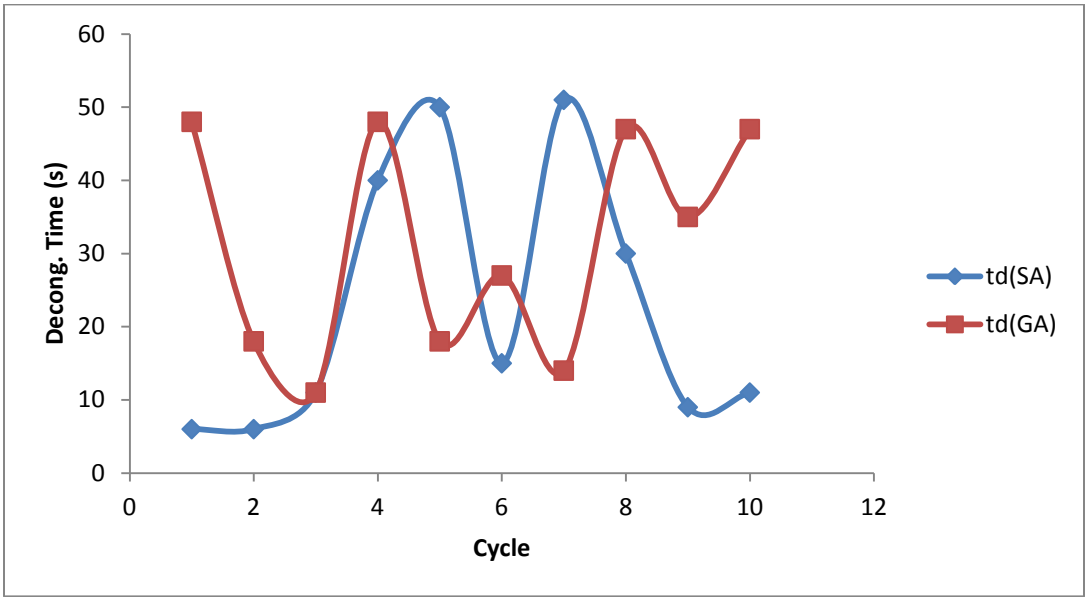

Fig. 5: Decongestion times in GA and SA for the Cycles

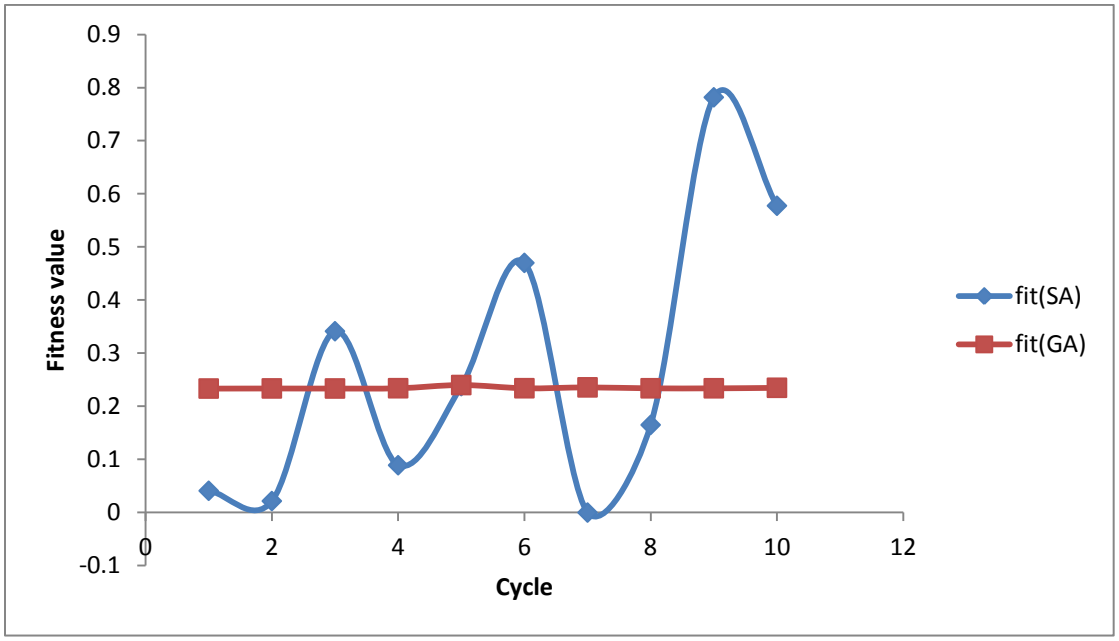

Fig. 6: Fitness values in SA and GA for the Cycles

With reference to Fig. 5, the period of decongestion under simulated annealing was higher as against the genetic algorithm considering their peak values of 51 for SA and 48 for GA. This probably confirms Mohammad (2008) and other researchers that support the declaration that SAs are usually slower than their contemporaries.

With reference to Fig. 6, it is observed that GA curve is almost linear, while that of SA is sinusoidal, which also probably shows that the convergence of SA is slower to GA; hence, corroborating Mohammad [4] as highlighted in the literature.

IV. DISCUSSION OF RESULTS

The Comparison of the optimization methods GA and SA to enhance Road Traffic Management is presented here. The control parameter values for all the optimization algorithms are given thus:

i. GA: Population size = 50, generations = 100, crossover probability = 0.05, mutation probability = 0.5.

ii. SA: Generations = 300, initial cooling temperature = 100 and cooling constant = 1, lower bound = -5 and upper bound = 5.

From the simulation results, it was shown that simulated annealing does not outperform genetic algorithm (Fig. 5 and Fig. 6). Considering Table 1, when C = 1, the decongestion time for SA is higher geometrically in comparison to GA. Meanwhile at the second cycle (i.e., C = 2), it was observed that the value generated for decongestion time declined sharply (by one third) and the highest time is seen at C = 4. By checking the fitness value at C =1 and 2, it was observed that there is an inverse proportion between the decongestion time and fitness function and irregular in some. Likewise, the value of the fitness functions generated were almost negligible (i.e., <<<0) in the simulated annealing result. Equal decongestion time was at C=9 and 10. Despite getting equal decongestion time at C = 9 and 10, different fitness values were recorded. Then, it is sufficient to say that there are trade-offs between decongestion time and fitness value in SA. Hence, the simulated annealing parameters – temperature, quenching factor and probabilities are responsible for these; probably confirming the result of Ola *et al*. [17].

On the other hand, with GA, the lowest decongestion time was seen at Cycle 3, and equal time was seen at C = 1 and 4 (Table II). The fitness values generated were also in the same range (0.2000 to 0.2357). The theory on convergence is proven that GA tends to converge faster than SA. Considering the relationship between the decongestion times and fitness values, there is no linear or inverse relationship as a result of the genetic algorithm method. The number of initial population and genetic operators and the pseudorandom numbers were responsible for these. The comparison and contrast between the variables in question were observed following the behaviour in terms of decongestion time and fitness values. In general, it was observed that the selection was random, and that when a lane was not fully decongested, another lane would not be picked. During the entire traffic decongestion cycle, values were recorded. However, the user needs to record the data at each cycle. Comparing the two cases, the fitness function statistics for the Case two (GA) was almost linear as against its counterpart (SA). The fitness values obtained were between 0 and 1 in all, which was between the selection probabilistic values set in the GA process. On the decongestion time, the values recorded showed that SA probably converges slower compared to Genetic Algorithm (Fig. 5). Also, idle processor and starving of one lane or the other as all lanes were touched in the ten (10) cycles.

A) Further Statistical Analysis

In order to confirm the results, we subjected the results obtained from SA and GA to statistical analysis. The independent t-test was used because of the two approaches. After setting the error value to 0.05 (95%), the analysis showed that there was no statistical significant difference between their fitness values ($p = 0.280 > 0.05$) as well as their decongestion ($p = 0.658 > 0.05$), even though there was mean difference between the two techniques.

Therefore, even though there are mean differences between the two techniques, there is no statistical significant difference between their performances.

V. CONCLUSION

In this study, Simulated Annealing (SA) and Genetic Algorithm (GA) models to solve the problem of road traffic congestion were simulated and compared. The simulations were implemented with MATLAB software. The traffic queue involved generation of pseudorandom numbers to substitute for the number of vehicles in the real-world scenario. The intersection consisted of a cross-road and an adjoining T-junction. The performance evaluation of the two approaches (GA and SA) was done. Ordinarily, the simulation results showed that GA outperformed SA; hence, corroborating Mohammad [4], Jukka [18] and Tarek [19]. Statistically, however, there were no differences in their performances.

Considering the need to improve the model, future research should apply animation. This will help add more understanding and attraction to the model. This could be in form of moving vehicles rather than the array of numbers as reflected in the current work. Also, there should be consideration of priority amongst motorist, such as Ambulance, Fire-fighting vehicle and Police van. The initial and stopping temperatures should be varied alongside cycle number for the Simulated Annealing program. For the Genetic Algorithm area, future researchers should consider the possibility of varying the initial population sizes, just like the cycle numbers. These would allow researchers to confirm their behaviour with respect to input size.

REFERENCES

[1] Kirkpatrick S., Gelatti C.D. and Vecchi P. (1983). Optimisation by Simulated Annealing, *Science Journal*, Series 220 Volume 4598 (May 13, 1983), pp. 671-680

[2] Dimitris Bertsimas and John Tsitsiklis (1993). Simulated Annealing, *Journal of Statistical Science*, Vol. 8 No. 1, pp. 10-15, Massachusetts

[3] Jeff Heaton (2008). Introduction to Neural Networks with Java, Heaton Research, Inc. (2nd ed.), United States of America, ISBN 1604390085, pp.199-207

[4] Mohammad Tehranipoor H. (2008). Simulated Annealing (SA) and CAD Algorithms, ECE Department, University of Connecticut, United States, *Journal paper on International Conference on Computer Aided design* (ICCAD).

[5] Johnson D., Aragon C., McGeoch L., and Schevon C. (1990). Optimisation by Simulated Annealing: An experimental evaluation, Part I: Graph partitioning. *Operations Research* 37, pp.865-892.

[6] Johnson, D. Aragon, C., McGeoch, L. and Schevon, C. (1991). Optimisation by Simulated Annealing: An experimental evaluation, Part II: Graph Colouring and Number Partitioning. *Operations Research* 39 878-406.

[7] Johnson, D. Aragon, C., McGeoch, L. and Schevon, C. (1992). Optimisation by Simulated Annealing: An experimental evaluation, Part III: The Travelling Salesman Problem. Unpublished manuscript.

[8] Sitharama S. Iyengar and Richard R. Brooks (2012). Distributed Sensor Networks, Second Edition: Image and Sensor Signal Processing, *Computer and Information Science Series*, Chapman and Hall/CRC Pres, Second Edition, ISBN 9781439862872, pp. 1-942

[9] Goldberg D. E. (1989). Genetic Algorithms in search, Optimization and Machine Learning, Addison-Wesley Publishing company, Inc., pp.1-145

[10] Holland J.H. (1992). Adaptation in Natural and Artificial Systems, MIT Press., Adaptation in Natural and Artificial Systems, *Complex Adaptive Systems Series*, Bradford Book Publishers, ISBN 0262581116, pp. 1-211.

[11] Steven Bergen and Brian J. Rose (2011). Automatic and Interactive Evolution of Vector Graphics Images with Genetic Algorithms. Technical Report to Department of Computer Science, Brock University, Canada, .pp. 2-14.

[12] Mühlenbein H. (1991). Evolution in Time and Space - The Parallel Genetic Algorithm, In Foundations of Genetic Algorithms, Morgan Kaufmann publishers, United States, pp.3-20.

[13] Whitley Darrell (1994). A Genetic Algorithms Tutorial, The GENITOR Research Group Publication on Statistics and Computing, Colorado State University, Fort Collins, Vol. 4, pp.65-85.

[14] Gustaf Jansson (2010). Traffic Control With Standard Genetic Algorithm: A Simulated Optimisation Control of a Traffic Intersection, Master Of Science Thesis/Thesis Work, In Intelligent Systems Design, Department of Applied Information Technology, Chalmers University of Technology, Gothenburg, Sweden, 2010, ISSN: 1651-4769, pp. 1-70

[15] Luke Brian .T. (2002). Simulated Annealing Flow Chart, Luke, B.T. & Associates Inc. Accessed from http://www.btluke.com/simanf2.html, December 29, 2015, pp. 1.

[16] Kenneth A. De Jong (1992). Are Genetic Algorithms Function Optimisers? *Proceedings of the Second Conference on Parallel Problems Solving from Nature*, Elsevier Publishers Ltd., Vol. 20, pp. 1-3.

[17] Ola B. O., Omidiora E. O., Ganiyu R. A. (2014). Modeling and Controlling Isolated Intersections using Simulated Annealing Approach, *International Journal of Applied Information Systems* (IJAIS) – Volume 7– No.6, ISSN : 2249-0868, pp. 13-15 .

[18] Jukka Kohonen (1999). A Brief Comparison of Simulated Annealing and Genetic Algorithm Approaches, Term paper for the "Three Concepts: Utility" course, Department of Computer Science, University of Helsinki, pp.1-3.

[19] Tarek (2007). A Genetic and Simulated Annealing Based Algorithms for Solving the Flow Assignment Problem in Computer Networks, *World Academy of Science, Engineering and Technology Journal*, Vol. 1, No 3, pp. 360-366.