



## A Performance Evaluation of Selected Heuristics for the Travelling Salesman Problem

<sup>1,2</sup>Okediran O. O., <sup>2</sup>Oyeleye C. A., <sup>3</sup>Adeyemo I. A.

<sup>1,2</sup>Department of Computer Science & Engineering, <sup>3</sup>Department of Electronics & Electrical Engineering,  
<sup>1,2,3</sup>Ladoke Akintola University of Technology, P.M. B. 4000, Ogbomoso, Nigeria

**Abstract:** *One of the classical problems in graph theory, which still has no close practicable exact algorithmic solution, is the Travelling Salesman's Problem. It is a NP complete problem whose solution space explodes exponentially as the number of nodes (cities) increases. Hence, recourse is often made to the use of heuristics to solve problems modeled after the travelling salesman problem. Heuristics have proved over the years to be very good feasible solution methods for solving combinatorial optimization problems such travelling salesman problem, vehicle routing problem, knapsack problem, graph coloring and so on. In this paper, ant colony optimization, genetic algorithm and simulated annealing were implemented and compared for solving some TSP instances. These three heuristics are widely used heuristics for solving combinatorial optimization problems.*

**Keywords:** *Travelling Salesman Problem, Ant Colony Optimization, Genetic Algorithm and Simulated Annealing.*

### I. INTRODUCTION

The traveling salesman problem, hereafter abbreviated and referred to as TSP, is a very well known NP-complete combinatorial optimization problem and is one of the most widely researched problems in Computer Science. The goal is to find the shortest tour that visits each city in a given list exactly once and then returns to the starting city [6, 7]. The TSP is to find the shortest Hamiltonian cycle in a graph. TSP can be phrased as a graph  $G$  whose vertices correspond to a set  $(C_1, C_2, C_3, \dots, C_N)$  of cities in the salesman's district and whose edges, that is distance  $d(C_i, C_j)$ , correspond to the roads joining two cities. The problem is to find an ordering of the cities that minimizes the quantity below:

$$G = \sum_{i=1}^{N-1} d(C_i, C_{(i+1)}) + d(C_N, C_1) \quad (1)$$

This quantity is referred to as tour length, since it is the total distance the salesman would make when visiting the cities in the order specified by the permutation, returning at the end to the initial city. If the distance (time and cost) between every pair of cities is independent of the direction of travel the problem ((the distances between the cities are independent of the direction of traversing the arcs, that is,  $d_{ij} = d_{ji}$  for every pair of nodes) is said to be symmetrical. If the distance between one or more pair of cities varies with the direction (at least for one pair of nodes  $i, j$  we have  $d_{ij} \neq d_{ji}$ ), the problem is called asymmetrical [5]. TSP has various applications in: combinatorial data analysis, computer wiring, machine sequencing, vehicle routing and scheduling, planning and logistics, Very large scale integrated circuit chip fabrication, X-ray crystallography and so on [1,10]. The importance of the TSP is that it is representative of a larger class of problems known as combinatorial optimization problems. The TSP problem belongs in the class of combinatorial optimization problems known as NP-complete [8].

Many exact algorithms have been written to solve this combinatorial problem and the execution time of these algorithms is however too large and even in most cases the computational resources cannot handle such algorithms. Consequently, there is a need to get an alternative means of solving these combinatorial problems. When it is impracticable to compute an optimal solution, there is the need to settle for good but not necessarily optimal solutions. Such solution procedures are called heuristic methods or simply heuristics.

### II. LITERATURE REVIEW

#### A. ANT COLONY OPTIMIZATION (ACO)

The ACO heuristic has been inspired by the observation on real ant colony's foraging behavior, and on that ants can often find the shortest path between food source and their nest. The principle of this phenomenon is that ants deposit a chemical substance (called pheromone) on the ground, thus, they mark a path by the pheromone trail. An ant encountering a previously laid trail can detect the dense of pheromone trail. It decides with high probability to follow a shortest path, and reinforce the trail with its own pheromone. The larger amount of the pheromone is on a particular path, the larger probability is that an ant selects that path and the path's pheromone trail will become denser. At last, the ant colony collectively marks the shortest path, which has the largest pheromone amount. Such simple indirect communication way among ants embodies actually a kind of collective learning mechanism [3, 11].

In ant system, each ant is initially put on a randomly chosen city and has a memory which stores the partial solution it has constructed so far (initially the memory contains only the start city). Starting from its start city, an ant iteratively moves from city to city. When being at a city  $i$ , an ant  $k$  chooses to go to a still unvisited city  $j$  with a probability given by

$$P_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} \quad \text{if } j \in N_i^k \quad (2)$$

where  $\eta_{ij} = 1/d_{ij}$  is a priori available heuristic information,  $\alpha$  and  $\beta$  are two parameters which determine the relative influence of pheromone trail and heuristic information, and  $N_i^k$  is the feasible neighborhood of ant  $k$ , that is, the set of cities which ant  $k$  has not yet visited. Parameters  $\alpha$  and  $\beta$  have the following influence on the algorithm behavior; if  $\alpha=0$  the selection probabilities are proportional to  $[\eta_{ij}]^\beta$  and the closest cities will more likely be selected: in this case AS corresponds to a classical stochastic greedy algorithm (with multiple starting points since ants are initially randomly distributed on the cities). If  $\beta=0$  only pheromone amplification is at work: this will lead to the rapid emergence of a stagnation situation with the corresponding generation of tours which, in general, are strongly suboptimal [2].

The solution construction ends after each ant has completed a tour, that is, after each ant has constructed a sequence of length  $n$ . Next, the pheromone trails are updated. In ant system this is done by first lowering the pheromone trails by a constant factor (this is pheromone evaporation) and then allowing each ant to deposit pheromone on the arcs that belong to its tour [4]:

$$\tau_{ij}(t+1) = (1 - \rho) \cdot \tau_{ij}(t) + \sum_{k=1}^m \Delta \tau_{ij}^k(t) \quad \forall (i, j) \quad (3)$$

where  $0 < \rho \leq 1$  is the pheromone trail evaporation rate and  $m$  is the number of ants. The parameter  $\rho$  is used to avoid unlimited accumulation of the pheromone trails and enables the algorithm to “forget” previously done bad decisions. On arcs which are not chosen by the ants, the associated pheromone strength will decrease exponentially with the number of iterations.  $\Delta \tau_{ij}^k(t)$  is the amount of pheromone ant  $k$  deposits on the arcs; it is defined as

$$\Delta \tau_{ij}^k(t) = \begin{cases} 1/L^k(t) & \text{if arc } (i, j) \text{ is used by ant } k \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where  $L^k(t)$  is the length of the  $k^{\text{th}}$  ant’s tour. By Equation 3, the shorter the ant’s tour is, the more pheromone is received by arcs belonging to the tour. In general, arcs which are used by many ants and which are contained in shorter tours will receive more pheromone and therefore are also more likely to be chosen in future iterations of the algorithm [4].

## **B. GENETIC ALGORITHM (GA)**

Genetic algorithm (GA) is a search and optimization method which works by mimicking the evolutionary principles and chromosomal processing in natural genetics. GA begins its search with a random set of solutions usually coded in binary strings. Every solution is assigned a fitness which is directly related to the objective function of the search and optimization problem. Thereafter, the population of solutions is modified to a new population by applying three operators similar to natural genetic operators-reproduction, crossover, and mutation. It works iteratively by successively applying these three operators in each generation till a termination criterion is satisfied [12]. Algorithmically, the basic genetic algorithm (GAs) is outlined as below:

- i. [Start] Generate random population of chromosomes, that is, suitable solutions for the problem.
- ii. [Fitness] Evaluate the fitness of each chromosome in the population.
- iii. [New population] Create a new population by repeating following steps until the new population is complete.
  - [Selection] Select two parent chromosomes from a population according to their fitness. Better the fitness, the bigger chance to be selected to be the parent.
  - [Crossover] With a crossover probability, cross over the parents to form new offspring, that is, children. If no crossover was performed, offspring is the exact copy of parents.
  - [Mutation] With a mutation probability, mutate new offspring at each locus.
  - [Accepting] Place new offspring in the new population.
- iv. [Replace] Use new generated population for a further run of the algorithm.
- v. [Test] If the end condition is satisfied, stop, and return the best solution in current population.
- vi. [Loop] Go to step 2.

## **C. SIMULATED ANNEALING (SA)**

Simulated Annealing is a probabilistic heuristic for the global optimization problem of locating a good approximation to the global optimum of a given function in a large search space. It is often used when the search space is discrete (for example all tours that visit a given set of cities). For certain problems, simulated annealing may be more

efficient than exhaustive enumeration provided that the goal is merely to find an acceptably good solution in a fixed amount of time, rather than the best possible solution. The name and inspiration come from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. Both are attributes of the material that depend on its thermodynamic free energy. Heating and cooling the material affects both the temperature and the thermodynamic free energy. While the same amount of cooling brings the same amount of decrease in temperature it will bring a bigger or smaller decrease in the thermodynamic free energy depending on the rate that it occurs, with a slower rate producing a bigger decrease [9]. This notion of slow cooling is implemented in the Simulated Annealing algorithm as a slow decrease in the probability of accepting worse solutions as it explores the solution space.

### III. METHODOLOGY

ACO, GA and SA were implemented to solve standard TSP instances downloaded from TSPLIB[13]. The code is in Matrix Laboratory (Matlab) and executed on windows 7 platform on a computer system with Intel core i5 processor and 4 GB RAM.

#### A. Pseudo code for ACO

When food is located, real Ants initially roams randomly from their colony to food. They deposit special chemical 'pheromone' on their path from colony to food. They also deposit some amount of pheromone while returning. Thus the ants following shortest path return earlier and amount of pheromone on that path is more. So after certain time this path has the more traffic because this is the shortest path. The Pheromones is evaporated at certain rate. However, longer paths which are not used by ants are eliminated after sometime. Thus ants using the history in terms of pheromone trail to search shortest path from colonies to their food. ACO algorithm is employed to imitate the behaviour of real ants as follows:

```
begin procedure ACO  
generate pheromone trails and other parameters  
while(termination criteria not meet)  
{  
construct solutions  
update pheromone Trails  
}  
post-process results and output  
end ACO procedure
```

In implementing the ACO algorithm in this work, the number of ants  $m$  is set to 10 while the parameters  $\alpha$  and  $\beta$  described above were set to 0.1 and 2 respectively.

#### B. Pseudo code for GA

GA uses the population of Chromosomes as the starting point then each chromosome is tested against fitness using an appropriate fitness function. Then the best chromosomes are selected and they undergo process of crossover and mutation to create new set of chromosome. This is depicted in the pseudo code below.

```
begin procedure GA  
generate populations and fitness function  
evaluate population  
while(termination criteria not meet)  
{  
while (best solution not meet)  
{  
crossover  
mutation  
evaluate  
}  
}  
post-process results and output  
end GA procedure
```

In implementing the GA in this work, population, mutation rate and cut length were set to 1000, 0.1 and 0.2 respectively.

#### C. Pseudo code for SA

In SA, each set of a solution represents a different internal energy of the system. Heating the system, results in a relaxation of the acceptance criteria of the samples taken from the search space. As the system is cooled, the acceptance criteria of samples are narrowed to focus on improving movements. Once the system has cooled, the configuration will represent a sample at or close to a global optimum. The pseudo code for SA is as follows:

```
begin procedure SA  
generate initial solutions  
set temperature, and cooling rate
```

```

while(termination criteria not meet)
{
generate new solutions
access new solutions
if (accept new solution)
{
update storage
adjust temperature
}
}
post-process results and output
end SA procedure

```

In implementing the SA algorithm in this work, temperature, cooling rate and absolute temperature were set to 10000, 0.9999 and 0.0001 respectively.

#### IV. RESULTS

The tables below shows TSP file names, number of cities and length of the optimal tour for ACO, GA and SA. The results given in the tables are best, worst and average of twenty runs for the three heuristics.

It was observed in the course of the implementation of the ACO algorithm that average speed to find the optimal solution defined as the inverse of the average time to find the optimal solution is a function of the number of ants in the ant colony. Also, higher communication trails increase the probability of finding optimal solution quickly. For GA and SA, there was a noticeable trade-off between runtime and solution quality. Summarily, results obtained showed that ACO and GA provided close results when the number of cities under consideration is few. However, GA, generally provided a better solution when compared ACO and SA.

Table 1: Performance of ACO for TSP

TSP Files	Number Of Cities	ACO		
		Best	Worst	Average
city10	10	128.34	135.47	127.95
city15	15	175.00	181.50	177.62
city29	29	8914.85	9970.25	9437.44
city30	30	524.96	555.40	543.31
city35	35	378.00	391.30	385.65
att48	48	3426.80	3793.15	3611.97
eil51	51	428.00	451.35	436.84
city51	51	502.72	556.34	526.45
eil75	75	542.30	551.03	548.56

Table 2: Performance of GA for TSP

TSP Files	Number Of Cities	GA		
		Best	Worst	Average
city10	10	111.96	117.10	112.35
city15	15	138.30	145.00	143.72
city29	29	8019.05	8154.20	8089.56
city30	30	463.00	491.50	475.90
city35	35	285.76	297.46	294.13
att48	48	2818.24	3008.30	2910.95
eil51	51	420.20	443.28	433.43
city51	51	414.00	427.14	419.05
eil75	75	533.60	537.81	534.75

Table 3: Performance of SA for TSP

TSP Files	Number Of Cities	SA		
		Best	Worst	Average
city10	10	140.24	148.90	143.04
city15	15	190.20	198.05	197.78
city29	29	17112.50	20245.31	18755.17
city30	30	588.52	602.58	594.38
city35	35	390.85	401.28	399.23
att48	48	2934.72	3614.40	3276.58

eil51	51	1102.64	1295.30	1200.35
city51	51	1252.15	1381.94	1318.85
eil75	75	598.26	605.79	607.02

## V. CONCLUSION

TSP is representative of a larger class of problems known as combinatorial optimization problems. The TSP belongs in the class of combinatorial optimization problems known as NP-complete. Specifically, if one can find an efficient algorithm (that is, an algorithm that will be guaranteed to find the optimal solution in a polynomial number of steps) for the traveling salesman problem, then efficient algorithms could be found for all other problems in the NP-complete class. To date, however, no polynomial-time algorithm can be guaranteed for the TSP. In other word, the travelling salesman problem cannot be effectively solved with exact algorithms hence the need for heuristics which have shown to be good TSP solvers. This study implemented the ant colony optimization algorithm, genetic algorithm and simulated annealing algorithm for solving standard TSP instances. To a great extent, the three heuristics are very good solvers of TSP and can guarantee optimal solutions. However, GA, generally provided a better solution when compared ACO and SA. As part of future work, two or more of these heuristics can be hybridized and the resulting heuristic's performance be compared with ACO, GA and SA for TSP.

## REFERENCES

- [1] Aybars U., Serdar K., Ali C., Muhammed C. and Ali A (2009), "Genetic Algorithm Based Solution of TSP on a Sphere, Mathematical and Computational Applications", Vol. 14, No. 3, pp. 219-228.
- [2] Dorigo M., (1992), "Optimization, Learning and Natural Algorithms, PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, pp.140.
- [3] Dorigo M., Maniezzo V., and Colormi A. (1996), " Ant System Optimization by a Colony of Cooperating Agents". IEEE Transactions on System, Man, Cybernetics. Part B, 26(1):pp.29-41.
- [4] Dorigo M. and Stutzle T. (2008), "The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances", 6<sup>th</sup> international conference, ANTS 2008, Brussels, Belgium.
- [5] Grotschel M. and Padberg M. W. (1985), "Polyhedral Theory: The Traveling Salesman Problem", Lawler, Lenstra, Rinnooy Kan and Shmoys, eds., John Wiley, pp 251-306.
- [6] Gutin G. and Punnen A. (2002), "The Traveling Salesman Problem and Its Variations", vol. 12 of Combinatorial Optimization. Kluwer, Dordrecht.
- [7] Hahsler M. and Hornik K. (2009),"Introduction to TSP- Infrastructure for the Traveling Salesperson Problem".
- [8] Hoffman K. and Padberg M. (2015), "Travelling Salesman Problem" available at <http://uran.donetsk.ua/masters/2009/kita/aleksandrova/library/article08.htm>. Accessed on 3<sup>rd</sup> May, 2015.
- [9] Kirkpatrick S., Gelatt C. D., Vecchi M. P. (1983) "Optimization by Simulated Annealing" Science 220(4598):671-680.
- [10] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., and Shmoys, D. B. (1985), "The Traveling Salesman Problem", John Wiley & Sons, Chichester.
- [11] Li Y. and Xu Z. (2003),"An Ant Colony Optimization Heuristic for Solving Maximum Independent Set Problems", Proceedings of the Fifth International Conference on Computational Intelligence and Multimedia Applications.
- [12] Malhotra R., Singh N. and Singh Y. (2011), "Genetic Algorithms: Concepts, Design for Optimization of Process Controllers", Journal of Computer and Information Science Vol. 4, No. 2.
- [13] TSBLIB (<http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html>)