International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 4– No.4, October 2012 – www.ijais.org

A Comparative Study of Simulated Annealing and Genetic Algorithm for Solving the Travelling Salesman Problem

Adewole A.P.
Department of Computer Science, University of Lagos, Akoka, Nigeria

Otubamowo K.
Department of Computer Science, University of Lagos, Akoka, Nigeria

Egunjobi T.O.
Department of Computer Science, University of Lagos, Akoka, Nigeria

ABSTRACT

Metaheuristic algorithms have proved to be good solvers for the traveling salesman problem (TSP). All metaheuristics usually encounter problems on which they perform poorly so the programmer must gain experience on which optimizers work well in different classes of problems. However due to the unique functionality of each type of meta-heuristic, comparison of metaheuristics is in many ways more difficult than other algorithmic comparisons. In this paper, solution to the traveling salesman problem was implemented using genetic algorithm and simulated annealing. These algorithms were compared based on performance and results using several benchmarks. It was observed that Simulated Annealing runs faster than Genetic Algorithm and runtime of Genetic Algorithm increases exponentially with number of cities. However, in terms of solution quality Genetic Algorithm is better than Simulated Annealing.

General Terms

Metaheuristics and Algorithms

Keywords

Genetic Algorithm, simulated Annealing, Travelling Salesman Problem, Candidate solution, Optimization problem.

1. INTRODUCTION

In computer science, metaheuristic refers to a method of computation that optimizes a problem by trying to improve a candidate solution with regard to a given measure of quality iteratively. Metaheuristics make few or no assumptions about the problem being optimized and are able to search very large spaces of candidate solutions. Generally, metaheuristics do not guarantee that an optimal solution is ever found, however deep knowledge of these metaheuristics can help produce near optimal results.

The Travelling Salesman Problem is an optimization problem which has various applications such as: combinatorial data analysis, computer wiring, machine sequencing, vehicle routing and scheduling, planning and logistics. These optimization problems can be effectively solved using various approaches that have been created. This paper examines simulated annealing algorithm and genetic algorithm as they are being used to solve the traveling salesman problem (TSP), which is a widely known combinatorial optimization problem in operations research and theoretical computer science. The concept of the traveling salesman problem is to seeking a tour of a specified number of cities (visiting each city exactly once and returning to the starting point) where the length of the tour is minimized [8]. The traveling salesman problem has many applications, from VLSI chip fabrication [4] to X-ray crystallography [14].

An optimization problem consists: an objective function and a set of constraints on variables. The task is to find the values of the variables that give an optimum value for the objective function, while satisfying all the constraints. The objective function may be a linear function in the variables or a nonlinear function in the variables and it may be to find the minimum value for the objective function. If for instance, the objective function represents the cost of or to find the maximum value of a profit function. The resources shared by the products and their manufacturing process are usually in limited supply or have some other restrictions on their availability. This consideration leads to the specification of constraints for the problem.

Each constraint usually takes the form of an equation or an inequality. The left side of such an equation or an inequality is an expression in the variables for the problem, and the right is a constant. The constraints may be linear or nonlinear depending on whether the expression on the left is a linear function or nonlinear function of the variables.

A linear programming problem is an optimization problem with a linear objective function as well as a set of linear constraints. An integer linear programming is a linear programming problem where the variables are required to have integer values. A nonlinear optimization problem has one or more constraints and/or the objective function is nonlinear.

Simulated Annealing is a probabilistic method used for obtaining the overall minimum of a function that may possess several local minima [3]. It is a Monte-Carlo maximization or minimization technique used for complex problems with many parameters and constraints. It mimics the process of annealing, which starts with a high temperature mix of metal and slowly cools the mix, allowing optimal structures to form as the material cools. The Simulated Annealing procedure randomly generates a large number of possible solutions, keeping both good and bad solutions. As the simulation progresses, the requirements for replacing an existing solution or staying in the pool becomes stricter and stricter, mimicking the slow cooling of metallic annealing. Eventually, the process yields a small set of optimal solutions. Simulated Annealing's advantage over other methods is its ability to obviate being trapped in local minima. This means that the algorithm does not always reject changes that decrease the objective function or changes that increase the objective function according to its probability function:

$$p = \exp\left(\frac{-\Delta f}{T}\right)$$

Where T is the control parameter (analogy to temperature) and Δf is the variation in the objective function. The probability function is a derivative of the Boltzmann probability distribution function [1].

Genetic Algorithm is a technique used for estimating computer models based on methods adapted from the field of genetics in biology. To use this technique, one encodes possible model behaviors into "genes". After each generation, the current models are rated and allowed to mate and breed based on their fitness. In the process of mating, the genes are exchanged, and crossovers and mutations can occur. The current population is discarded and its offspring forms the next generation. Also, Genetic Algorithm describes a variety of modeling or optimization techniques that claim to mimic some aspect of biological modeling in choosing an optimum. Typically, the object being modeled is represented in a fashion that is easy to modify automatically. Then a large number of candidate models are generated and tested against the current data. Each model is scored and the "best" models are retained for the next generation. The retention can be deterministic (choose the best k models) or random (choose the k models with probability proportional to the score.) These models are then randomly perturbed (as in asexual reproduction) and the process repeated until it converges. If the model is constructed so that they have "genes," the winners can "mate" to produce the next generation.

2. TRAVELLING SALESMAN PROBLEM

The TSP is an NP complete problem, it is probably the most widely studied combinatorial optimization problem. It is a conceptually simple problem which is useful for solving several real life optimization problems [2]. A Classical Traveling Salesman Problem (TSP) can be defined as a problem where starting from a node it is required to visit every other node only once in a way that the total distance covered is minimized. This can be mathematically stated as follows:

$$\text{Min:} \quad \sum_{i,j} c_{ij}\, x_{ij} \qquad (1)$$

$$\text{s.t:} \quad \sum_{j} x_{ij} = 1;\ \forall\, i \neq j \qquad (2)$$

$$\sum_{i} x_{ij} = 1;\ \forall\, j \neq i \qquad (3)$$

$$u_i = 1 \qquad (4)$$

$$2 \leq u_i \leq n;\ \forall\, i \neq 1 \qquad (5)$$

$$\left\{ \begin{array}{c} u_i - u_j + 1 \leq \\ (n-1)(1 - x_{ij}) \\ \forall\, i \neq j, \forall j \neq i \end{array} \right\} \qquad (6)$$

$$u_i \geq 0; \forall i \qquad (7)$$

$$x_{ij} \in \{0,1\};\ \forall i, j \qquad (8)$$

Constraints set (4), (5), (6) and (7), are used to eliminate any sub tour in the solution. Without the additional constraints for sub tour elimination, the problem reduces to a simple assignment problem, which can be solved as an L.P. (Linear Programming) without binary constraints on xij and will still result in binary solution for xij. Introduction of additional constraints for sub tour elimination, however, makes the problem an M.I.P. (Mixed Integer Problem) with n2 integer variables for a problem of size n, which may become very difficult to solve for a moderate size of problem [17].

3. METHODOLOGY

Comparison of the algorithms is based on dataset, solution quality, parameters and runtime.

3.1 Data Set/Test Bed

One of the most important parts of a comparison among heuristics is the test bed on which the heuristics are tested. Consequently, the test bed should be first considered when comparing two metaheuristics [12]. Existing test beds from previous papers will be used alongside some other new testbeds.

Various test beds are used, a new geometric city was constructed, a distance matrix of some major Nigerian cities was gotten from www.bosng.org, some were fetched from data used in previous works and the others are generated randomly by the program.

3.2 Parameters

Parameters are the configurable components of an algorithm that can be changed to alter the performance of the algorithm. They can either be set statically (for instance, creating a genetic algorithm with a population size of 50) or based on the problem instance (for instance, creating a genetic algorithm with a population size of $5\sqrt{n}$, where n is the number of nodes in the problem instance) [12]. In either of these cases, the algorithm designer predetermines the function of the problem instance attributes used to generate the parameter or the constant value of the parameter.

A number of parameters must be set for each major type of metaheuristic before algorithm execution. These guidelines represent the minimum number of parameters typical in different types of algorithms. However, most metaheuristics have more parameters in practice. Table 1 below shows the parameters used for each of the two algorithms being compared.

Table 1. Parameters used for Simulated Annealing and Genetic Algorithm.

Genetic Algorithm	Simulated Annealing
Population	Temperature
Mutation Rate	Cooling Rate
Cut Length	Absolute Temperature

3.3 Run Time

While it is important that a metaheuristic demonstrate good solution quality in order to be considered viable, having a fast runtime is also of utmost necessity. If the runtime of metaheuristics is not fast, it would not be justifiable to choose them over exact algorithms. Also, runtime comparisons are some of the most difficult comparisons to make. This is because of the difficulties in comparing runtimes of algorithms that are compiled with different compilers and executed on different computers, potentially on different testbeds [12].

3.4 Solution Quality

Although it is essential collect a meaningful testbed and to juxtapose the metaheuristics in terms of simplicity by considering their number of parameters, equally more important among the comparisons is the solution quality. Meta-heuristics give solutions of good quality in runtimes better than those of exact approaches. The amount of permissible deviation from the optimal solution varies according to the application. For example, in many long-term planning applications or applications critical to an organisation's business plan the amount of error allowed is much lower than in optimization problems used for short-term planning or for which the solution is tangential to an organisation's business plans. Also, when considering the same problem, the amount of error allowed can differ dramatically. For example, a parcel company planning its daily routes to be used for the next year using the capacitated vehicle routing problem would likely have much less error tolerance than a planning committee using the capacitated vehicle routing problem to plan the distribution of voting materials in the week leading up to Election Day.

Consequently, determining a target solution quality for a combinatorial optimization problem is often difficult/impossible. Thus, it is not sufficient to determine if each heuristic meets a required solution quality threshold when comparing metaheuristics; comparison among the heuristics is required [12].

4. IMPLEMENTATION

4.1 Simulated Annealing Program Details

A TSP class was created which has 4 methods and 15 instance variables. The methods and their functions are explained below.

4.1.1 Openfile()

This method initializes currentOrder and nextOrder and then displays a JFileChooser that lets you browse for the text folder that contains the distances matrix. It also reads the file and sets disances[i][j] according to the text folder.

4.1.2 GetTotalDistance()

Calculates the cost of a tour which is input as a parameter in form of a list.

4.1.3 GetNextArrangement()

Finds a new tour that does not exist previously by randomization

4.1.4 Anneal()

The iterations are carried out here, new tours are generated continuously based on previous tours until no more visible exist in the new tours created.

In the interface the button 'load' triggers the method openFile and also causes the distances matrix to be displayed on the JTextArea labeled input. The other button on the form which is the 'solve' button calls the method Anneal() which solves the problem loaded previously and displays the result on the second JTextArea labeled output.

4.2 Genetic Algorithm Program Details

In genetic algorithm, class "Chromosome" is needed. The Chromosome class generates random tours and makes them population members when its object is instantiated in the TSP class. The TSP class uses the Chromosomes "mate" method to reproduce new offspring from favoured Population of the previous generations. The TSP class in this case has two methods that use methods in Chromosome, the two methods are described below.

4.2.1 Start()

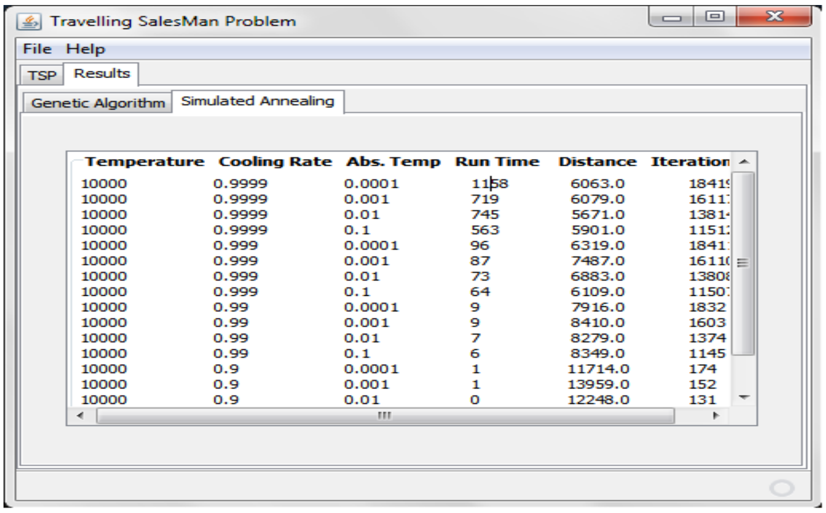
This method initializes the cities and creates new chromosomes by creating an array of Chromosome objects, it also sorts the chromosomes by calling the method sortChromosomes() in Chromosomes then it sets the generation to 0

4.2.2 Run()

Gets the favoured population from all the chromosomes created and mates them using mate() after this it sorts the chromosomes and then calculates the cost of the tour of the best chromosome. It repeats this this procedure until the cost of the best tour can't be further improved.

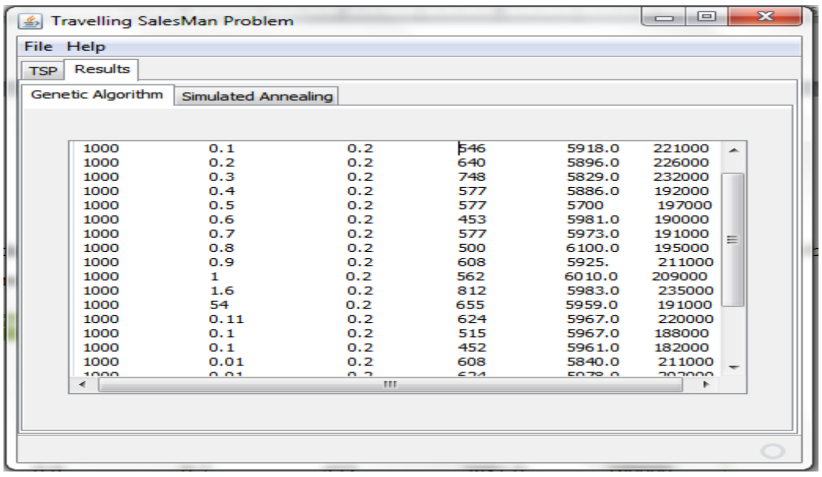
5. RESULTS AND DISCUSSIONS

The result of each trial is stored in a text area so as to keep track of results. There is a tab for each algorithm, the screen shots below shows the experimental results of Simulated Annealing and genetic algorithm respectively.


Travelling SalesMan Problem
File Help
TSP Results
Genetic Algorithm Simulated Annealing

Temperature	Cooling Rate	Abs. Temp	Run Time	Distance	Iteration
10000	0.9999	0.0001	1158	6063.0	1841
10000	0.9999	0.001	719	6079.0	1611
10000	0.9999	0.01	745	5671.0	1381
10000	0.9999	0.1	563	5901.0	1151
10000	0.999	0.0001	96	6319.0	1841
10000	0.999	0.001	87	7487.0	1611
10000	0.999	0.01	73	6883.0	1380
10000	0.999	0.1	64	6109.0	1150
10000	0.99	0.0001	9	7916.0	1832
10000	0.99	0.001	9	8410.0	1603
10000	0.99	0.01	7	8279.0	1374
10000	0.99	0.1	6	8349.0	1145
10000	0.9	0.0001	1	11714.0	174
10000	0.9	0.001	1	13959.0	152
10000	0.9	0.01	0	12248.0	131

Figure 1: Experimental results for Simulated Annealing


Travelling SalesMan Problem
File Help
TSP Results
Genetic Algorithm Simulated Annealing

1000	0.1	0.2	546	5918.0	221000
1000	0.2	0.2	640	5896.0	226000
1000	0.3	0.2	748	5829.0	232000
1000	0.4	0.2	577	5886.0	192000
1000	0.5	0.2	577	5700	197000
1000	0.6	0.2	453	5981.0	190000
1000	0.7	0.2	577	5973.0	191000
1000	0.8	0.2	500	6100.0	195000
1000	0.9	0.2	608	5925.	211000
1000	1	0.2	562	6010.0	209000
1000	1.6	0.2	812	5983.0	235000
1000	54	0.2	655	5959.0	191000
1000	0.11	0.2	624	5967.0	220000
1000	0.1	0.2	515	5967.0	188000
1000	0.1	0.2	452	5961.0	182000
1000	0.01	0.2	608	5840.0	211000

Figure 2: Experimental results for Genetic Algorithm

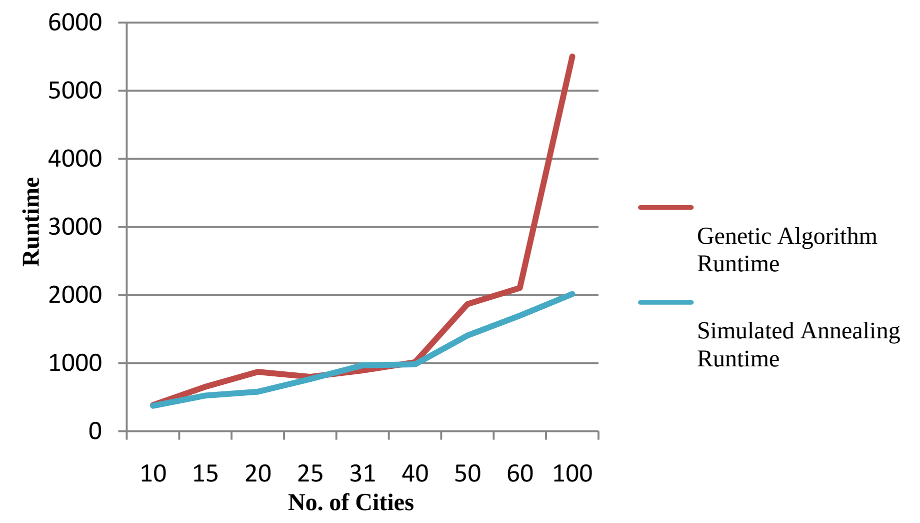
The application keeps track of results for analysis by storing the result of each problem and the parameters set for each run. The screen shots above (figure 1 and 2) display the result of random combination of parameters to get the parameters with best performance (table 2). This was done for Genetic Algorithm and Simulated Annealing. Random experiments on the 31 Nigerian cities have been used to determine the best set of parameters for both algorithms.

Table 2. Parameters with best performance.

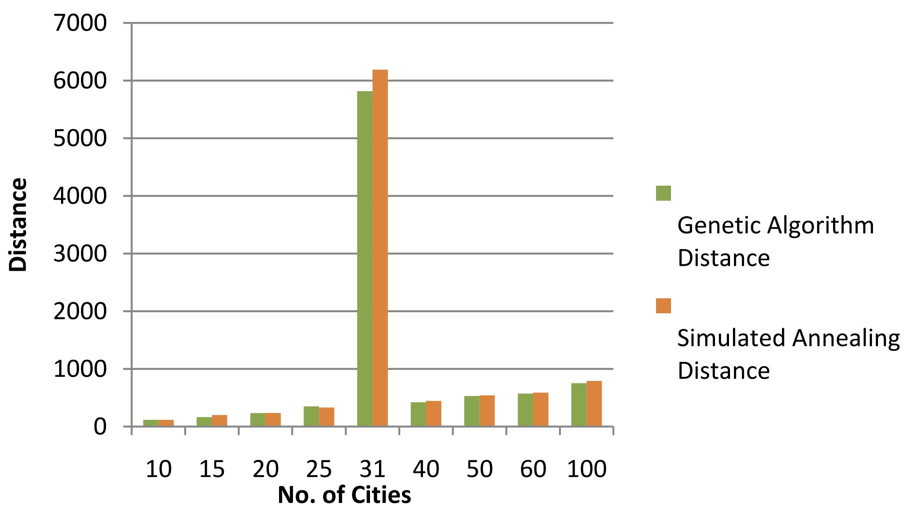
Simulated Annealing	**Genetic Algorithm**
Temperature: 10000	Population: 1000
Cooling Rate: 0.9999	Mutation Rate: 0.1
Absolute Temperature: 0.0001	Cut Length: 0.2

Table 3 Results obtained using the parameters above for different test beds.

No. of	Genetic Algorithm			Simulated Annealing		
Cities	Runtime	Distance	Individuals	Runtime	Distance	Iterations
10	385	118.0	148000	373	116.0	184197
15	653	163.0	174000	523	199.0	184197
20	871	233.0	256000	580	236.0	184197
25	799	351.0	278000	765	330.0	184197
31	893	5818.0	177000	967	6189.0	184197
40	1014	423.0	334000	982	444.0	184197
50	1867	530.0	329000	1405	543.0	184197
60	2105	572.0	366000	1698	589.0	184197



Graph 1: Plot of Runtime against No. of Cities



Graph 2: Bar Chrart Showing distance gainst No. of Cities

The results obtained are discussed below,

5.1 Runtime
From graph 1 above (which was plotted from the values in table 3), the line representing Genetic algorithm shows that time complexity increases exponentially with increasing number of cities. For fifty cities and above, the algorithm takes a lot of time to complete (about 20000ms or more).

In Simulated Annealing increase in number of cities increases the runtime but the increments are not exponential as is the case in genetic algorithm. The runtime is also affected by the set of parameters used. In genetic algorithm, small sized population can greatly improve the runtime but the solution quality is likely reduced depending on the number of cities. Therefore from the above assertions, if runtime is the basis of comparison of both algorithms, Simulated Annealing has proven to be a faster TSP solver. This is due to the discovery by [8] that: the simulated annealing algorithm is run for a total of 106 iterations on each trial while the genetic algorithm is run for a total of 105 iterations per trial. The variance in the number of trials is due to the fact that one iteration of genetic algorithm takes approximately 10 times longer than one iteration of simulated annealing.

5.2 Solution Quality
Again, the set of parameters used and their values is a very crucial factor that affects solution quality. In simulated annealing a moderately high temperature and cooling rate very close to 1 say 0.9999 produces near optimal if not optimal result. On the other hand a population that is greater than the number of cities divided mutation rate can produce very good results. Generally larger population provides better solution but the run time is greatly increased in Genetic Algorithm. In the long run genetic algorithm tends to produce better results for large number of cities if the run time is not an issue because from the result above and according to [8], genetic algorithm takes time to perform its iterations which greatly improves the quality of its solution; a major reason why it is slower compared with simulated annealing.

5.3 Test Beds
The test bed on which the heuristics are tested is important. The same testbeds must be used on for each method. Existing testbeds from previous researches were used in this research alongside some other new testbeds. A new geometric city was constructed because there are no sufficient existing testbeds comprising major Nigerian cities, a distance matrix of some major Nigerian cities was also gotten from www.bosng.org and the others are generated randomly by the program. This allowed for large problem instances to be tested. The solution of small problem instances runs in reasonable runtimes with the assurance of a guaranteed optimal solution but it is important that metaheuristic testing occurs on large problems for which optimal solutions could not be calculated in reasonable runtimes.

5.4 Parameters
The set of parameters used is very crucial to the performance of the algorithms. For genetic algorithm, when the population size is set to 1000, the mutation rate set to 0.1 and the cut rate at 0.2 yields an optimal solution for the TSP.

For simulated annealing, a temperature of 10000, cooling rate of 0.9999 and absolute temperature of 0.0001 work best to solve the TSP problem optimally. Different classes of problems require unique set of parameters that provides results which have a good balance of runtime and solution.

6. CONCLUSION
In this paper the Travelling Salesman problem was extensively discussed with the aim of finding out which of the two algoriths being compared would yield an optimal solution. The travelling salesman problem cannot be effectively solved with exact algorithms hence the need for metaheuristics which have shown to be good TSP solvers. Examples of metaheuristics are Ant Colony Optimization, Tabu Search, Genetic Algorithms, Simulated Annealing and so on. This study implemented the genetic algorithm and the simulated annealing algorithm and also to gain experience on how to use them by observing their performance on different test beds. Experiments have shown that Simulated Annealing has a faster runtime. Simulated Annealing can solve up to 350 cities with a 2.16GHz processor within a reasonable amount of time.

Genetic Algorithms however can provide quality solutions if a large enough population is set, but large population greatly increases its run time. Powerful systems with parallel computing capability can however be of help here and can be used to run these algorithms for very large no of cities.

To a great extent, both algorithms are very good solvers and can provide optimal solutions if the right set of parameters are set. If solution quality is important, cooling rate that is very close to one should be set for simulated annealing but this increases the no of iterations the algorithm will perform. For a tour of less than 500, an Absolute temperature of 10000 is recommended. For Genetic Algorithm the larger the population the higher the possibility of getting an optimal solution but run time increases exponentially. Generally there exists a broad trade-off between runtime and solution quality.

7. FUTURE STUDY
The techniques of comparison discussed in this study are: testbeds, parameters, solution quality and runtime. It was discovered that the runtime technique depended on invariants such as: compiler choice, programmer skill and power of computation platform [12]. This made the determination of a target solution quality for a combinatorial optimization problem difficult. Consequently, other forms of comparison that do not rely on the invariants above are desired. One alternative form as discussed by [1] is counting the number of representative operation that the algorithm uses. In this form, the numbers of a selected set of bottleneck operations are compared with disregard for the total execution time of algorithms being compared. This technique and others can be sought to use as a basis for comparison. The observations made from these new techniques of comparison can then be compared with results of comparison from this study to determine how they influence getting an optimal result for the TSP and checking whether they yield better outputs compared to the ones yielded in this study. It must however be stated that every comparison technique has its strengths and weaknesses.

8. REFERENCES

[1] Ahuja, R., Orlin, J. 1993. Use of representative operation counts in computational testing of algorithms. INFORMS Journal on Computing 8(3), 318-330

[2] Ali Hamdar 2008. Simulated Annealing-Solving the Travelling Salesman Problem.

[3] Anthony Ralston and Edwin D. Reilly 1993. Encyclopedia of Computer Science, Chapman & Hall.

[4] 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.

[5] Bertsimas D. and Tsitsiklis J. 1993. "Simlated Annealing", Journal of Statistical Science, Vol. 8, No 1, 10-15.

[6] Bland, R. G. and Shallcross, D. F. 1989. Large Traveling Salesman Problems Arising from Experiments in X-ray Crystallography: A preliminary report on computation, Operations Research. Letter. pg.125-128.

[7] Dorigo M., Vittorio M. and Alberto C. 1996. The Ant System: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, MAN, and cybernetic, vol. 26, No. 1.

[8] Fan Yang 2010. Solving Traveling Salesman Problem Using Parallel Genetic Algorithm and Simulated Annealing.

[9] Hiroaki Sengoku and Ikuo Yoshihara 1993, A Fast TSP solver using GA on JAVA.

[10] Holland, J. H. 1992, Adaptation in Natural and Artificial systems, Cambridge, MA, USA: MIT Press.

[11] John Silberholz and Bruce Golden 2010, Comparison of Metaheuristics.

[12] Kirkpatrick, S., Gelatt, Jr. C.D. and Vecchi, M.P. 1983. "Optimization by Simulated Annealing", Journal of Mathematical Sciences, Vol. 220: pg. 109-120.

[13] Korte, B. 1988. Applications of Combinatorial Optimization, talk at the 13th International Mathematical Programming Symposium, Tokyo.

[14] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., and Shmoys, D. B. 1985. The Traveling Salesman Problem, John Wiley & Sons, Chichester.

[15] Mark Dorigo and Thomas Stutzle(2009), Ant Colony Optimization.

[16] Nazif, H. and Lee, L.S. 2010. "Optimized Crossover Genetic Algorithm for Vehicle Routing Problem with Time Windows, American", Journal of Applied Sciences 7 (1): pg. 95-101.

[17] Roland Braune, Stefan Wagner and Michael Affenzeller 2005, Applying Genetic Algorithms to the Optimization of Production Planning in a real world Manufacturing Environment, Institute of Systems Theory and Simulation Johannes Kepler University.

[18] Sachin Jayaswal 2004. A Comparative study of Tabu Search and Simulated Annealing for Travelling Salesman problem.

[19] TSP2010. http://www.tsp.gatech.edu/history/tspinfo

[20] Wikipedia 2010. The Free Encyclopedia, http://en.wikipedia.org/wiki/Main_Page.

[21] Zuhaimy Ismail, Wan Rohaizad and Wan Ibrahim 2008. "Travelling Salesman problem for solving Petrol Distribution using Simulated-Annealing", American Journal of Applied Sciences 5(11): 1543-1546.