International Journal of Intelligent Computing Research (IJICR), Volume 6, Issue 4, December 2015

Performance Comparison of Simulated Annealing, GA and ACO Applied to TSP

Hosam H. A. Mukhairez, Ashraf Y. A. Maghari
Faculty of Information Technology, Islamic University of Gaza
Gaza, Palestine

Abstract

The travelling salesman problem (TSP) is probably one of the most famous problems in combinatorial optimization. There are many techniques to solve the TSP problem such as Ant Colony Optimization (ACO), Genetic Algorithm (GA) and Simulated Annealing (SA).In this paper, we conduct a comparison study to evaluate the performance of these three algorithms in terms of execution time and shortest distance. JAVA programing is used to implement the algorithms using three benchmarks on the same platform conditions. Among the three algorithms, we found out that the Simulated Annealing has the shortest time in execution(<1s) but for the shortest distance, it comes in the second order. Furthermore, in term of shortest distance between the cities, ACO performs better than GA and SA. However, ACO comes in the last order in term of time execution.

1. Introduction

Optimization is one of the most important tasks of engineers, which the engineer always asked to design more efficient and less expensive systems as well as to devise such plans and techniques to improve operations of running systems in many fields especially in industrial and the scientific world. The travelling salesman problem (TSP) is a nondeterministic polynomial hard problem in combinatorial optimization studied in algorithms and operations research ,also theoretical computer science studies.[1]

The core problem mainly summarized as there are cities and given costs, weights or distances between them, a travelling salesman required to visit all cities, but he want to save time on travelling, therefore we need to find the suitable sequence of cities to minimize the traveled costs, weights or distances. [1]

A salesman decided to travel to M different cities. The most existing important question appears is: In what advised continues list of cities should he visit to minimize the total distance traveled or cost? Each city is expressed as a letter (e.g. 'A' or 'B').

If we have M cities, and we want to compute all paths between them, then the possible combinations or sequences of cities are M factorial. For example, 30 cities produce 30! Combinations which equals 2.65252859812191058636308 48e+32. This is a very big number of probabilities and combinations. If we tried every combination of sequences and could test 10,000 of those sequences per second it would take at normal computers more than 8 million years to randomly get and observe the minimum sequence.

In 1930 the problem was presented as a mathematical problem and considered as one of the most intensively studied problems in optimization. It is used as a benchmark for many optimization algorithms. Even though the problem is computationally difficult, a large number of heuristics and exact methods are known, so that some instances with tens of thousands of cities can be solved.[1]
Alhanjouri and AlFarra presented the TSP in some real world actions:

- Arranging school buses routes to pick up students.
- Delivering meals to people at homes.
- Scheduling stacker cranes in a warehouse,
- Planning truck routes to pick up parcel post and others materials.
- Planning, logistics, and the manufacture of microchips.
- A classic example of the TSP is the scheduling of a machine to drill holes in a circuit board.

There are many approaches and algorithms for solving TSP problem such as Dijkstra, Minimum Spanning Tree, and Nearest Neighbor (NNH), Ant colony optimization (ACO), Genetic Algorithm (GA), and Simulated Annealing (SA). This study conducts a comparison between ACO, GA, and SA. The comparison between them is accomplished to state the better one for solving travelling salesman problem.

Ant colony optimization (ACO) is one of the old and most popular meta-heuristics used for combinatorial optimization (CO) in which an optimal solution is sought over a discrete search space. The well-known CO's issue is the traveling salesman problem (TSP) where the search of suitable candidate solutions grows up relatively to the increase size of the problem, which leads to almost infeasible optimal solution appears.

The first ACO algorithm –Ant System (AS) - has been introduced by Marco Dorigo in the early 1990's, and several developments of the AS have been suggested [2];[3]. The ACO algorithm is based on a computational paradigm informed from real ant colonies and the way they work. The idea was to use several constructive computational agents (simulating real ants).

Ant's behavior is dominated by the goal of colony existence rather than being interested on the existence of individuals. The behavior that provided the inspiration for ACO is the ants' seeking behavior (see figure 1), and in particular, how ants can find shortest paths between food places and their nested camp. When searching for food, ants initially discover the area around their nest in a random way. While moving around, each ant leaves a chemical pheromone trail on the ground. Other ants can smell pheromone.

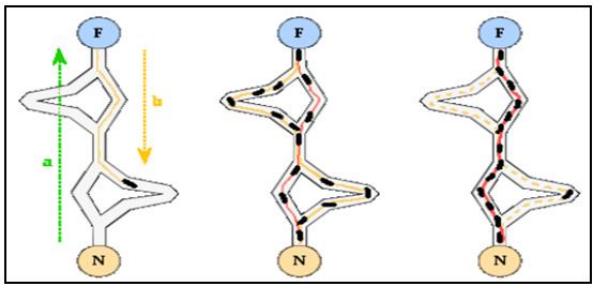

Figure 1. Ants use pheromone as indirect communication to build best tour [1]

When choosing their way, they look to choose, in probability, paths scored by strong pheromone concentrations. When any ant finds a food source, it weighs up the quantity and the quality of the food and carries some of it back to the nest. While returning trip, the quantity of pheromone that an ant leaves on the ground may depend on the quantity and quality of the food. The pheromone marks will guide other ants to the food source. It has been shown that the indirect messages between the ants through pheromone enables them to find shortest paths between their nested camps and food sources.

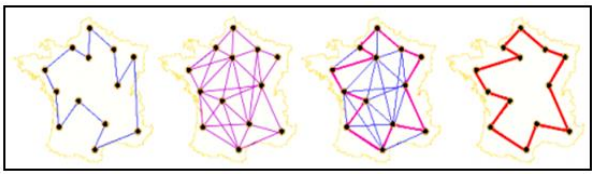

Figure 2. Undirected graph show nodes and edges, the figure show the four stages of ACO to reach shortest path [1]

Genetic algorithms are a part of evolutionary computing technique, which is a rapidly growing area of artificial intelligence. Genetic algorithms are inspired by Darwin's theory about development and evolution. Rechenberg said: "solution to a problem solved by genetic algorithms is evolved". [4] displayed the idea of evolutionary computing in the 1960s in his work "Evolution strategies" (Evolutions strategy in original). After that researchers developed on his idea. Genetic Algorithms (GA) were designed and planned by John Holland and developed by him and his students and colleagues. In 1992 Koza [5] has used GA to advance programs to perform defined tasks. He called his method "genetic programming" (GP), LISP programs were used; because programs in this language can be expressed in the form of a "parse tree", which is the object the GA works on.

Basic Explanation of GA

Genetic algorithm is started with a set of solutions (denoted by chromosomes) called population. Solutions from one population are booked and used to form a new population. This is motivated by a hope, that the new generation will be better than the old one in its characteristics.

Solutions which are selected to form new solutions (offspring) are selected according to their fitness attributes; the more suitable they are the high probability they have to replicate. This action is repeated until certain conditions are satisfied.

Outline of the basic Genetic Algorithm

1. (Start) Generate random population of n chromosomes (suitable solutions for the problem).
2. (Fitness) Evaluate the fitness f(x) of each chromosome x in the population.
3. (New population) Create a new population by repeating the following steps until the new population is complete.

a. (Selection) Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected).

b. (Crossover) with a crossover probability cross-over the parents to form a new offspring (children). If no

crossover was performed, offspring is an exact copy of parents.
c. (Mutation) with a mutation probability mutate new offspring at each locus (position in chromosome).
d. (Accepting) Place new offspring in a new population.
4. (Replace) use new generated population for a further run of algorithm.
5. (Test) if the end condition is satisfied, stops, and returns the best solution in current population.
6. (Loop) Go to step 2.

The above outline of GA is very general. There are many things that can be implemented differently in various problems.

The simulated annealing (SA) algorithm was originally encouraged from the process of strengthening metal by heating. Strengthening involves heating and cooling a material to modify its physical properties due to the changes in its internal structure. As the metal cools its new structure becomes fixed, consequently causing the metal to retain its newly obtained properties.

In simulated annealing we keep a temperature variable to simulate this heating process. We initially set it high and then allow it to slowly 'cool' as the algorithm runs. While this temperature variable is high the algorithm will be allowed, with more frequency, to accept solutions that are worse than our current solution. This gives the algorithm the ability to jump out of any local optimums it finds itself in early on in execution. As the temperature is reduced so is the chance of accepting worse solutions, therefore allowing the algorithm to gradually focus in an area of the search space in which hopefully, a close to optimum solution can be found. This gradual 'cooling' process is what makes the simulated annealing algorithm remarkably effective at finding a close to optimum solution when dealing with large problems which contain numerous local optimums. The nature of the traveling salesman problem makes it a perfect example.

For doing the comparisons, TSPLIB95 corpus used, which includes hundreds of tsp maps that can be used as benchmarks. We selected three benchmarks which are (bier127.tsp, berlin52.tsp, ali535.tsp). Our comparison study concentrates on time execution of the algorithms, and the smallest shortest distance between cities.

The rest of this paper is organized as follows: Section 2 discusses state of the art and reviews some related works. Section 3 shows experimental setup. Section 4 presents results and discussion. Finally Section 5 the conclusion.

2. Overview and Related work

2.1. Ant Colony Optimization (ACO)

Dorigo and Gambardella described a qualified simulated ant colony for solving the travelling salesman problem (TSP). Ants of the simulated colony are able to generate one after another shorter feasible trips by using information gathered in the form of a pheromone trail dropped on the edges of the TSP graph. Computer simulations prove that the artificial ant colony is talented of producing good solutions to TSP. The method is an example, like simulated annealing, neural networks and evolutionary computation, of the successful use of a natural metaphor to design an optimization algorithm.[6]

Thomas and Marco [7] presented experimental solutions and results which have been obtained with MAX-MIN Ant System, which is one of the improved forms of Ant System.

2.2. Genetic Algorithms (GA)

The traveling salesman problem (TSP) is used as an idea model for a wide-range class of problems having complication due to the combinatorial explosion. The TSP has become a target for the genetic algorithm (GA) researchers, because it is probably the significant problem in combinatorial optimization and many new ideas in combinatorial optimization have been tested on the TSP.

However, by using GA for solving TSPs, Tsujimura and Gen, M obtained a local optimal solution rather than a best estimated solution frequently. The goal of their work is to solve (TSP) problem about local optimal solutions by announcing a degree of diversity of populations using the concept of information entropy. Thus, they obtained a best approximate solution of the TSP by using entropy-based GA. [8].

SENGOKU and YOSHIHARA developed a Java GUI software depends on a hybrid algorithm using GA and heuristics for quick solution of TSP[9], Also they claim that their TSP solver is useful as a criterion for assessing the performance of TSP solvers.

2.2. Hybrid approach of (GA) and (ACO)

Gong and Ruan proposed a hybrid method of genetic algorithm (GA) and ant colony optimization (ACO) for the TSP. In the proposed method, every chromosome of GA is also at the same time an ant of ACO. Whenever GA achieves the operation of crossover and mutation, the method firstly computes

the relationship strength between gene codes of parental chromosome(s) according to the pheromone matrix of ACO, and it then chooses the crossover or mutation point(s) according to the relationship strength. A threshold is produced to classify the gene relationship as strong or weak, the strong relationship segments of parents are reserved to offspring as far as possible [10].

Chunxiang and Xiaoni integrated genetic algorithm (GA) with ant colony optimization (ACO) for solving Traveling Salesman Problems(TSP) to get better optimization performance than each single algorithm alone, and complement advantages each other and get out of each other disadvantages. The hybrid algorithm (HA) runs GA first and then ACO. [11]

A new approach called GSA was proposed focusing at the key link in the hybrid algorithm (HA) that translates genetic solution from GA into information pheromone to distribute in ACO. GSA takes new matrix which is formed by the combination of the former 90% of individual from genetic solution and 10% of individual by random generation as the basis of transformation of pheromone value. They also discussed the best combination of genetic operators in GA. Many TSP samples were used as modelling tests to test genetic operators matching and optimization performance of HA. The results showed that PMX crossover matched with IVM mutation in the GA is the best combination of genetic operators which is able to make GA improve the precision of optimal solution, and HA using the best combination operators and GSA approach is effective and available to search for finest solution in high efficiency and has good convergence. [11].

Also Shahla Nemat, Mohammad Ehsan Basiri and others[12] interested in combining GA with ACO to improve the performance of finding solution for TSP problem. They suggested a new feature selection algorithm that combines genetic algorithms (GA) and ant colony optimization (ACO) for faster and better search experience. Their hybrid algorithm makes use of advantages of both ACO and GA methods. Suggested algorithm is easily implemented and because of use of a simple classifier in that, its computational complexity is very low. The performance of suggested algorithm is compared to the performance of two prominent population-based algorithms, ACO and GA.

Experimentation is carried out using two challenging biological datasets, involving the hierarchical functional classification of GPCRs and enzymes.

2.2. Comparing between GA and ACO

Some other authors interested in comparing between GA and ACO in solving TSP problem, Haroun et al. [13] presented a contribution to comparing two nature inspired metaheuristics for solving the TSP. They run ACO and GA on three benchmark examples with changing size and complexity; also they run GA and ACO one real world application in the field of urban transportation and logistics.

They observed that the GA is fast, easy to implement and cost efficient in terms of computational resources. The ACO is greedier but gives better results, especially with large problems size.

Shuang et al. [14] proposed a hybrid PS-ACO algorithm, ACO algorithm altered by particle swarm optimization (PSO) algorithm. The pheromone updating rules of ACO are merged with the local and global search mechanisms of PSO, they claims that PS-ACO algorithm has better convergence performance than genetic algorithm (GA), ACO and MMAS under the condition of limited evolution iterations.

Alhanjouri and Alfarra [1] tried to apply both techniques to solve TSP by using the same dataset and compare between them to determine which one is better in solving travelling salesman problem. For Ant Colony Optimization, they studied the effect of some parameters on the generated results, these parameters as: number of Ants, evaporation, and number of iterations. On the other hand, they studied the mutation probability, chromosome population, and crossover probability parameters that effect on the Genetic Algorithm results. Their comparison between GA and ACO is achieved to form suitable algorithm for travelling salesman problem. At the end they observed that GA is still better than ACO for TSP.

2.3. Simulated Annealing (SA)

Yip and Pao[15] presented a new technique, which integrates the idea of simulated annealing into the practice of simulated evolution, in place of arbitrary heuristics. The presented technique is called guided evolutionary simulated annealing (GESA). Their results show that the GESA technique can discover a very good near finest solution after examining a very small fraction of possible solutions.

After reading and viewing many papers talks about GA, ACO and SA in solving TSP problem, we are going run some implementations using JAVA code to state which is better in the terms of (Best Distance, Execution Time) when running the algorithms on the same platform condition and datasets.

2.4. Comparing GA, ACO, and SA

Kumbharana and Pandey[16] compared between ACO, GA and SA for TSPLIB samples (city10, city29, city51, ulysses16, Oliver30, att48, eli51) in term of shortest distance using matlab. Their results showed best, worst and average distances of 15 executions for ACO, GA and SA.

However, this work is different in the context of assessing GA, ACO, and SA in terms of time execution in addition to shortest distance between cities. Furthermore, the algorithms have been ordered according to these two terms and our experiments were conducted using java programming.

3. Experimental results

All the simulations were completed on a Windows 10 64-bits laptop computer with an i7-4510U processor clocked at 2.00GHz, and 6 GB of Ram. The Genetic algorithm was developed in JAVA using "GAlib" library. The Ant colony optimization algorithm and Simulated Annealing was written in native JAVA code.

Algorithms GA, ACO and SA will be run in NetBeans8.1 and JDK1.8 using JAVA programming on the same *.tsp dataset files download from TSPLIB95 corpus.

3.1. The Ant Colony Optimization

Table 1. Basic ACO algorithm parameters values

Attribute	value
ALPHA	-0.2
BETA	9.6
PHEROMONE_PERSISTENCE	0.3
INITIAL_PHEROMONES	0.8
NUM_OF_ANTS	2048

3.1.1.Ali535.tsp. Ali535 consists of 535 city, ACO took many time to calculate the result. It found a tour of best shortest distance of 1510.637 in ~62 seconds.

```
Best Distance : 1704.9441419834145     124 ms
Best Distance : 1669.2584485372467     240 ms
Best Distance : 1635.463791954742     246 ms
Best Distance : 1602.1780632898583     382 ms
Best Distance : 1602.1664278794744     661 ms
Best Distance : 1570.7817210112046     934 ms
Best Distance : 1547.7942743401723     2081 ms
Best Distance : 1546.2615828623912     7644 ms
Best Distance : 1530.8317940586758     8872 ms
Best Distance : 1510.6378747251965     62346 ms
Finished
Found best so far: 1510.6378747251965
Took: 62346 ms!
```

Figure 3. Evolution of ACO results over time in ali535

3.1.2. Berlin52.tsp. Berlin52 consists of 52 city, ACO quickly calculated the result. It found a tour of best shortest distance of 7721.432 in ~4 seconds.

```
Best Distance : 9314.015228328099     18 ms
Best Distance : 9046.022193401845     21 ms
Best Distance : 8372.289273078923     22 ms
Best Distance : 8093.352348459832     55 ms
Best Distance : 7962.446369947605     280 ms
Best Distance : 7937.283328985864     350 ms
Best Distance : 7669.9935935435515     4376 ms
Finished
Found best so far: 7669.9935935435515
Took: 4376 ms!
```

Figure 4. Evolution of ACO results over time in berlin52

3.1.3. Bier127.tsp. Bier127 consists of 127 city, ACO took some time to return the result. It found a tour of best shortest distance of 124651.524 in ~27 seconds.

```
Best Distance : 162025.46431956737     56 ms
Best Distance : 156335.23540292942     56 ms
Best Distance : 142817.9583602962     64 ms
Best Distance : 138674.47744156013     78 ms
Best Distance : 137521.11531041062     184 ms
Best Distance : 132718.10845923843     282 ms
Best Distance : 132604.64856500865     353 ms
Best Distance : 127793.69417221849     378 ms
Best Distance : 125768.95713720501     1376 ms
Best Distance : 124651.52414312352     27202 ms
Finished
Found best so far: 124651.52414312352
Took: 27202 ms!
```

Figure 5: Evolution of ACO results over time in bier127

Table 2. Summary of ACO results over time for the three tsp maps

Map Name	Nods #	Time(s)
bier127	127	~27
berlin52	52	~4
ali535	535	~62

3.2. The Genetic Algorithm

Table 3. Basic GA algorithm parameters values

Attribute	Value
MUTATION RATE	0.06
TOUR SIZE	10
POPULATION SIZE	2000
POPULATION GENERATION	100

3.2.1 ali535.tsp. The GA gets results faster than ACO but doesn't found the optimal solution. It found a tour of best shortest distance of 9466 in ~5 second.

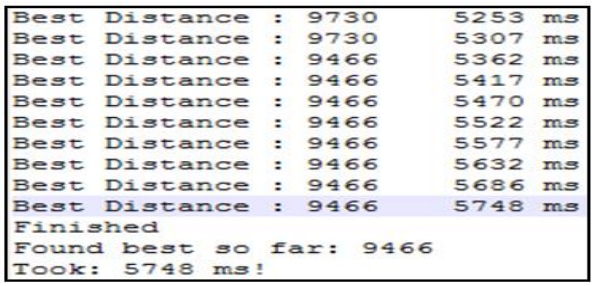

Figure 6. Evolution of GA results over time in ali535

3.2.2. Berlin52.tsp. Also in berlin52, the GA gets results of best distance of 11551 in ~1 second.

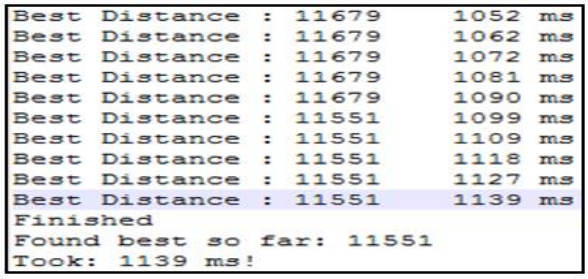

Figure 7. Evolution of GA results over time in berlin52

3.2.3. Bier127.tsp. Also in bier127, the GA gets results faster than ACO. It found a tour of best shortest distance of 419224 in ~3 second.

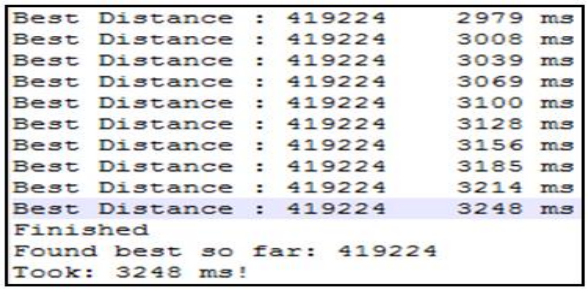

Figure 8. Evolution of GA results over time in bier127

Table 4. Summary of GA results over time for the three tsp maps

Map Name	Nods #	Time(s)
bier127	127	~3
berlin52	52	~1
ali535	535	~5

3.3. Simulated Annealing (SA)

Table 5. Basic SA algorithm parameters values

Attribute	Value
INITIAL TEMPRETUR	10000
COOLING RATE	0.003

3.3.1. Ali535.tsp. The SA returned the Final solution distance of 6471 in ~0.3 seconds.

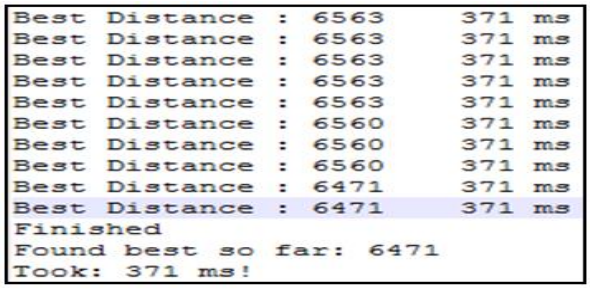

Figure 9. Evolution of SA results over time in ali535

3.3.2. Berlin52.tsp. The SA returned the Final solution distance of 10586 in ~0.2 seconds.

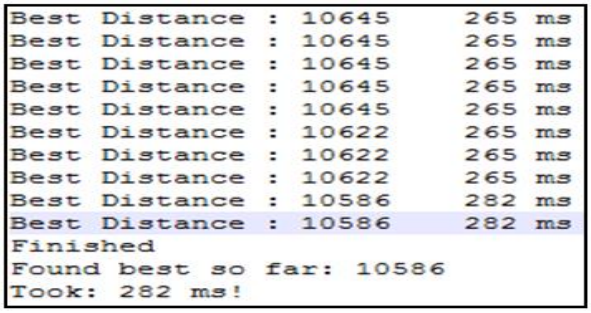

Figure 10. Evolution of SA results over time in berlin52

3.3.3. Bier127.tsp. The SA returned the Final solution distance of 265289 in ~0.3 seconds.

```
Best Distance : 265568      287 ms
Best Distance : 265568      287 ms
Best Distance : 265568      287 ms
Best Distance : 265568      287 ms
Best Distance : 265568      287 ms
Best Distance : 265568      287 ms
Best Distance : 265568      287 ms
Best Distance : 265568      287 ms
Best Distance : 265289      315 ms
Best Distance : 265289      315 ms
Finished
Found best so far: 265289
Took: 315 ms!
```

Figure 11. Evolution of SA results over time in bier127

Table 6. Summary of SA results over time for the three tsp maps

Map Name	Nods #	Time(s)
bier127	127	~0.3
berlin52	52	~0.2
ali535	535	~0.3

Table 7. Overview of simulation results

Instance	GA		ACO		SA	
	Best Distance	Time (s)	Best Distance	Time (s)	Best Distance	Time (s)
bier127	419224	~3	124651.524	~27	265289	~0.3
berlin52	11551	~1	7721.432	~4	10586	~0.2
ali535	9466	~5	1510.637	~62	6471	~0.3

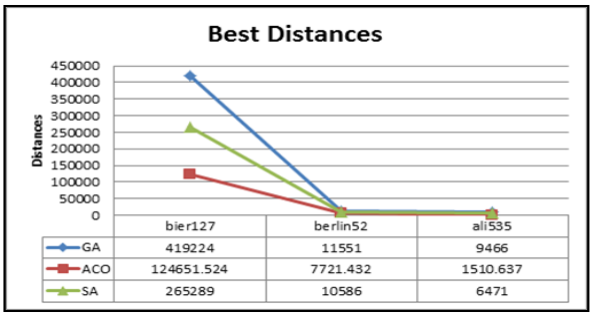

Figure 12. Evolution of GA, ACO and SA results over best distance for bier127, berlin52, and ali535

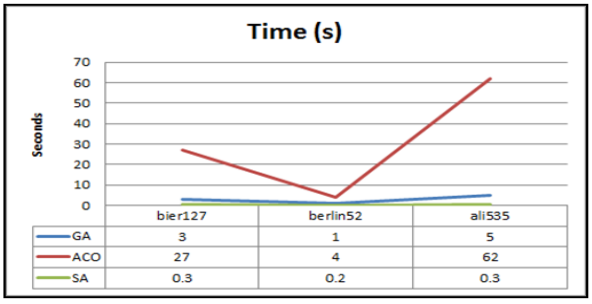

Figure 13. Evolution of GA, ACO and SA results over time(s) for bier127, berlin52 and ali535

4. Results and Discussion

Based on our simulation results we observed that:

- GA comes in the second order in both finding the shortest path and execution time. So it couldn't be considered the optimal algorithm for solving TSP.
- ACO comes in the first order in finding the shortest path, but it takes a long time in execution compared to other algorithms. So it could be considered suitable algorithm in finding optimal shortest distance solution between cities.
- SA comes in the first order in time execution (< 1s) and gives average shortest distance results between GA and ACO.
- All these algorithms may give variable results if they run on other platforms conditions or different attribute values of each of the compared algorithms such as (ALPHA, BETA, MUTATION RATE, POPULATION SIZE, INITIAL TEMPRETUR, COOLING RATE,…. etc.).

Result stated in point number 2 is consistent with finding made in [1] and [13], and is conflicting with finding made in [16].

5. Conclusion

This paper presented a comparative study view between most widely used optimization algorithm techniques Optimization (ACO, GA and SA) in terms of shortest distance and execution time.

Our goal was to evaluate the performance of these algorithms in terms of execution time and shortest distance under the same platform conditions. JAVA programing was used to implement the algorithms using three benchmarks.

We found out that the Simulated Annealing has the shortest execution time (< 1s) but it was at the second order for the shortest distance term among the compared algorithms. Furthermore, in term of shortest distance between the cities, ACO stated better than GA and SA. However, ACO appeared in the last order in term of execution time.

In the future – A combination between two of the compared approaches (SA & ACO) is suggested. Whereas ACO give the best shortest distance and SA saves time in execution. So they complement each other and cancel out their own limitations.

6. References

[1] Alhanjouri, M. and B. Alfarra, Ant Colony versus Genetic Algorithm based on Travelling Salesman Problem.

International Journal of Computer Technology and Applications, 2011. 2(3).

[2] Dorigo, M. and L. Gambardella. Ant-Q: A reinforcement learning approach to the traveling salesman problem. in Proceedings of ML-95, Twelfth Intern. Conf. on Machine Learning. 2014.

[3] Stützle, T. and H. Hoos. MAX-MIN ant system and local search for the traveling salesman problem. in Evolutionary Computation, 1997., IEEE International Conference on. 1997. IEEE.

[4] Rechenberg, I., Evolution strategy. Computational Intelligence: Imitating Life, 1994. 1.

[5] Koza, J.R., Genetic programming: on the programming of computers by means of natural selection. Vol. 1. 1992: MIT press.

[6] Dorigo, M. and L.M. Gambardella, Ant colonies for the travelling salesman problem. Biosystems, 1997. 43(2): p. 73-81.

[7] Stutzle, T. and M. Dorigo, ACO Algorithms for the Travelling Salesman Problems, Evolutionary Algorithms in Engineering and Computer Science, 1999, John-Wiley & Sons.

[8] Tsujimura, Y. and M. Gen. Entropy-based genetic algorithm for solving TSP. in Knowledge-Based Intelligent Electronic Systems, 1998. Proceedings KES '98. 1998 Second International Conference on. 1998.

[9] Sengoku, H. and I. Yoshihara. A fast TSP solver using GA on JAVA. in Third International Symposium on Artificial Life, and Robotics (AROB III'98). 1998.

[10] Daoxiong, G. and R. Xiaogang. A hybrid approach of GA and ACO for TSP. in Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on. 2004.

[11] Chunxiang, W. and G. Xiaoni. A hybrid algorithm based on genetic algorithm and ant colony optimization for Traveling Salesman Problems. in Information Science and Engineering (ICISE), 2010 2nd International Conference on. 2010.

[12] Nemati, S., et al., A novel ACO–GA hybrid algorithm for feature selection in protein function prediction. Expert Systems with Applications, 2009. 36(10): p. 12086-12094.

[13] Haroun, S.A., B. Jamal, and E.H. Hicham, A Performance Comparison of GA and ACO Applied to TSP. International Journal of Computer Applications, 2015. 117(20): p. 28-35.

[14] Shuang, B., J. Chen, and Z. Li, Study on hybrid PS-ACO algorithm. Applied Intelligence, 2011. 34(1): p. 64-73.

[15] Yip, P.P. and Y.-H. Pao, Combinatorial optimization with use of guided evolutionary simulated annealing. Neural Networks, IEEE Transactions on, 1995. 6(2): p. 290-295.

[16] Kumbharana, N. and G.M. Pandey, A Comparative Study of ACO, GA and SA for Solving Travelling Salesman Problem. International Journal of Societal Applications of Computer Science, 2013. 2(2): p. 224-228.

