International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 9 (2018) pp. 6808-6816

Performance Comparison of Genetic Algorithm, Particle Swarm Optimization and Simulated Annealing Applied to TSP

Madhumita Panda

Assistant Professor, Computer Science
SUIIT, Sambalpur University.
Odisha, India.

Abstract

The travelling salesman problem (TSP) is a well-known, popular and extensively studied problem in the field of combinatorial optimization. It is an optimization problem of finding a shortest closed tour that visits all the given cities within the shortest time. There are many techniques used to solve the TSP problem such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA).In this paper, a comparison study was made to evaluate the performance of these three algorithms in terms of execution time and shortest distance. JAVA programming is used to implement the algorithms using two benchmarks and one personal instance on the same platform conditions. Among the three algorithms, we found out that the Simulated Annealing has the shortest time in execution but for the shortest distance, it comes in the second order. Furthermore, in term of shortest distance between the cities, PSO performs better than GA and SA.

Keywords:-Travelling salesman problem (TSP), Genetic algorithm (GA), Particle swarm optimization (PSO), Simulated Annealing (SA)

INTRODUCTION

Travelling salesman problem (TSP) is one of the most famous problems in combinatorial optimization in which a salesperson intends to visit a number of cities exactly once, and return to its starting point, while minimizing the total distance travelled or the overall cost of the trip.(It belongs to the class of NP-hard optimisation problems [1],where the computational complexity increases exponentially wi th the number of cities. Thus, 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, 20 cities produce 20! Combinations which equals 2.432902e+18.TSP is a sub-problem of many applications such as transportation[2],wireless communications [3], vehicle routing [4], integrated circuits[5] and semiconductor industries [6].

No efficient algorithm exists for the TSP and all its relevant variants or problem of the same class. The need to quickly find good (not necessarily optimal) solutions to these problems has led to the development of various approximation algorithms such as the metaheuristics [7, 8]. Metaheuristic algorithms use search strategies to explore only promising part of the search space, but most effectively. Among the most popular metaheuristics, ,to name a few are, genetic algorithms (GA), simulated annealing (SA), ant colonies optimization (ACO) which are presented in [8], particle swarm optimization (PSO) [9],Bee colonies optimization (BCO) [10], monkey search algorithm (MS) [11],harmony search algorithm (HS) [12], firefly algorithm (FA) [13], intelligent water drops (IWD) [14], and cuckoo search [15] are among the recently proposed metaheuristics.

In [16], a GA-based algorithm has been introduced for TSP. It uses a topological encoding scheme to form ordering of vertices in a directed graph. The paper[17] provided a flexible method to solve the TSP using genetic algorithm. TSP is the main domain in the paper to solve the NP-hard problems. The referenced paper provides an implementation using genetic which is used to find shortest path for a specific problem. A modified particle swarm optimization was proposed in [18] to solve travelling salesman problem (TSP). The algorithm searched in the Cartesian continuous space, and constructed a mapping from continuous space to discrete permutation space of TSP, thus to implement the space transformation. Moreover, local search technique was introduced to enhance the ability to search, and chaotic operations were employed to prevent the particles from falling into local optima prematurely. Finally four benchmark problems in TSPLIB were tested to evaluate the performance of the algorithm. Experimental results indicate that the algorithm can find high quality solutions in a comparatively short time.Shi, *et.al*[19] modified the subtraction operator between two particle positions in PSO. The crossover eliminated technique and two local search techniques were also executed in their method. These local search techniques were used to accelerate the convergence speed, by employing the generalized chromosome.This paper[20]presents a hybrid discrete particle swarm optimization (HDPSO) for solving the travelling salesman problem (TSP). The HDPSO combines a new discrete particle swarm optimization (DPSO) with a local search. It is an approach designed for the TSP based on the binary version of particle swarm optimization. Unlike in general versions of particle swarm optimization, DPSO redefines the particle's position and velocity, and then updates its state by using a tour construction. The embedded local search is implemented to improve the solutions generated by DPSO. The experimental results on some instances are reported and indicate HDPSO can be used to solve TSPs. This paper[21] describes and analyses a novel distributed implementation of the simulated annealing algorithm to find a good solution to the travelling salesman problem. The implementation runs on a linear chain of processors driven by a host processor, which plays only a

supervisory role, so that the bulk of processing takes place on the chain and the efficiency of the algorithm remains high as the number of processors is increased.

This study conducts a comparison between GA, PSO, and SA to solve the travelling salesman problem in terms of execution time and shortest distance. Section 2 of the paper gives a brief discussion of the three optimization methods used in the study. Section 3 gives the experimental results and a comparison between the three algorithms (GA,PSO,SA) is accomplished to state the better one for solving travelling salesman problem. The conclusion section is presented at the end of this paper with future work.

OPTIMIZATION METHODS

The Genetic Algorithm(GA)

Genetic Algorithm was introduced by Holland et al. [22]. It is motivated by Darwin's theory about evolution and based on mimicking the survival of the fittest among the species generated by random changes in the gene-structure of the chromosomes in the evolutionary biology [23].

In the Genetic Algorithm logic, a solution vector is called individual or chromosome. Each chromosome is made of discrete units called genes and each gene controls one or more elements of the chromosome. Normally, a chromosome is a unique solution in the solution space. GA operates with a set of chromosomes, called population. The population is normally initialized randomly [23].

The basic process for a genetic algorithm is as follows:

1. **Initialization**: Generate random population of n chromosomes (suitable solutions for the problem).This population is usually randomly generated and can be any desired size, from only a few individuals to thousands.
2. **Evaluation**: Each member of the population is then evaluated and a 'fitness' value for that individual is calculated.
3. **Selection**: Discard the bad designs and keep only the best individuals in the population.
4. **Crossover**: With a crossover probability, crossover the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents. The hope is that by combining certain traits from two or more individuals, an even 'fitter' offspring which will inherit the best traits from each of its parents, will be created.
5. **Mutation**: With a mutation probability, mutate new offspring at each locus (position in chromosome).This is done to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next by making small changes at random to an individual's genome.
6. **Repeat**: After obtaining the next generation, start again from step two until reaching a termination condition

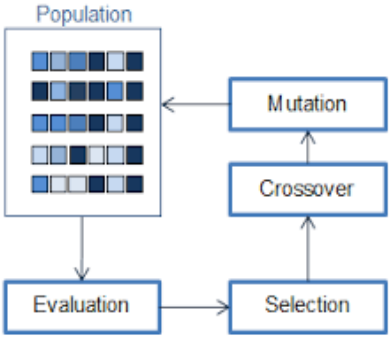

Figure 1. Genetic Algorithm

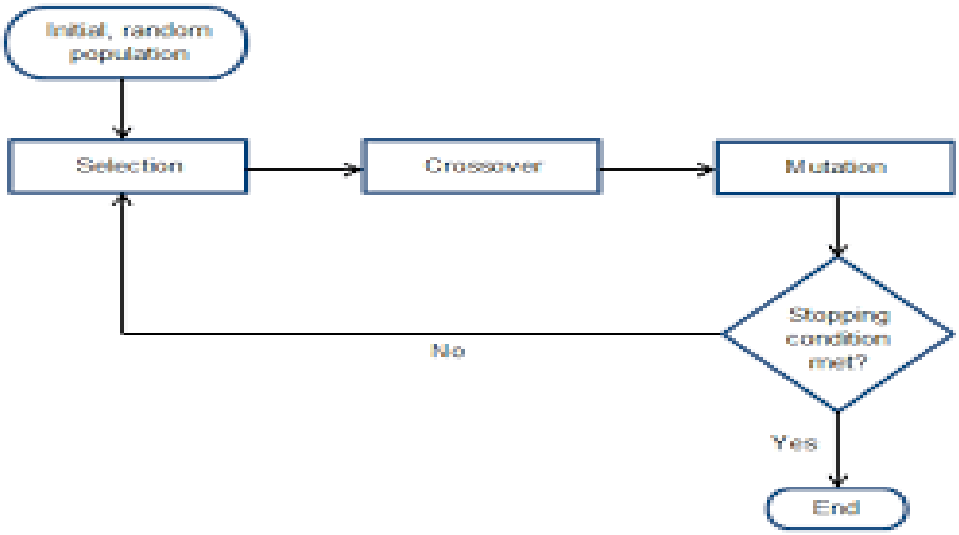

Figure 2. Flowchart of Genetic Algorithm

Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is an optimization technique introduced by Kennedy and Eberhart in 1995 [24]. It uses a simple mechanism that mimics swarm behaviour in birds flocking and fish schooling to guide the particles to search for global optimal solutions.

In PSO, each single solution is a "bird" in the search space which we call it "particle". All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles. PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. At each iteration, each particle is updated by the following two values: the first one is the best solution (fitness) it has achieved so far, which is called *pbest*. Another value that is tracked by the particle swarm optimizer is the best value obtained so far by any particle in the population. This best value is a global best and called *gbest*. When a particle takes part of the population as its topological neighbours, the best value is a local best and is called *lbest*. After finding the two best values, the particle updates its velocity and positions .

The formulas of PSO algorithm are as follows [24,25]:

$$v_{id}^{t+1} = v_{id}^{t} + c1.rand(0,1).(p_{id}^{t} - x_{id}^{t}) + c2.rand(0,1).(p_{gd}^{t} - x_{id}^{t}) \quad (1)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad (2)$$

Where v_{id}^{t} and x_{id}^{t} are particle velocity and particle position respectively. *d* is the dimension in the search space, *i* is the particle index, and *t* is the iteration number. c_1 and c_2 represent the speed, regulating the length when flying towards the most optimal particles of the whole swarm and the most optimal individual particle. p_i is the best position achieved so far by particle *i* and p_g is the best position found by the neighbours of particle *i*. *rand*(0,1) is the random values between 0 and 1. The exploration happens if either or both of the differences between the particle's best (p_{id}^{t}) and previous particle's position (x_{id}^{t}), and between population's all-time best (p_{gd}^{t}) and previous particle's position (x_{id}^{t}) are large, and exploitation occurs when these values are both small.

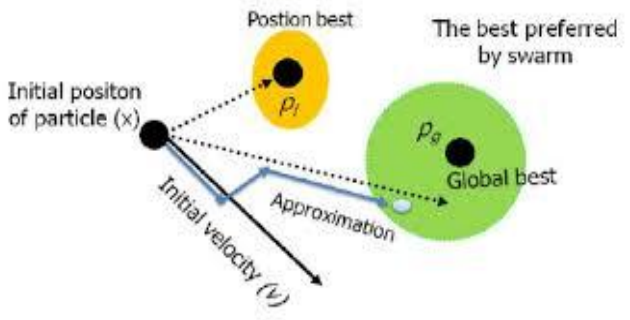

Figure 3. Particles movement

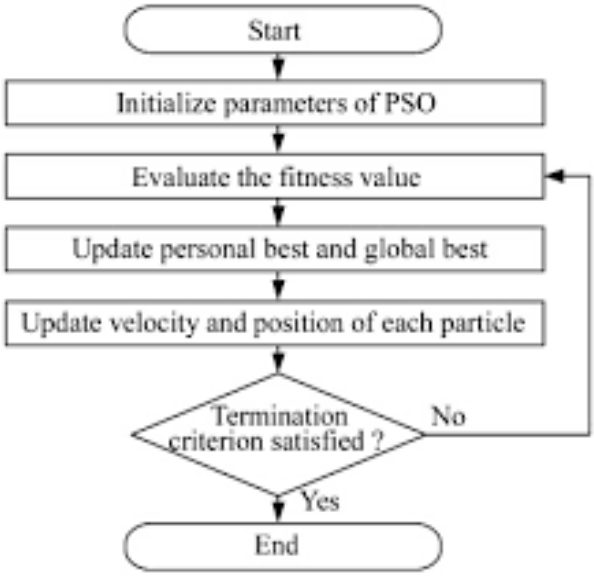

Figure 4. Flowchart of the PSO algorithm

Simulated Annealing (SA)

The SA is a meta-heuristic optimization method founded on the annealing process of metal re-crystallization[26]. If this process is allocated with enough time, SA could then find the optimal solution of a considered problem. Based on this analogy of how metal is cool and annealed, each step of the SA algorithm replaces the current solution by a random nearby solution which is gradually decreased during the searching process

System Design and Implementation

All the simulations were completed on a Windows8 64-bits laptop computer with an i3-2350M processor clocked at 2.30GHz, and 2 GB of Ram. All the three algorithms was developed in JAVA.

To test the algorithmic approaches, it was decided to run several experiments, varying the complexity of the TSP. The three algorithms were tested on two benchmark instances and one **personal instance** which coordinates are:

X {30, 50, 43, 40, 40, 36, 21, 8, 16, 7, 3, 80, 55, 60, 45, 1, 30, 45, 25, 3};

Y {5, 9, 10, 30, 20, 43, 48, 40, 36, 21, 18, 16, 27, 17, 80, 1, 12, 21, 65, 4};

TSPLIB: http://www.iwr.uniheidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html, provides most of the benchmark problems. It offers problem coordinates, best tour and the optimum solutions for the three test problems The benchmark TSPs used are **Berlin52,Att-48**.

EXPERIMENTAL RESULTS

(a)The Genetic Algorithm

Table 1. Basic GA algorithm parameters values

Genetic Algorithm	
Population Generation	100
Tour Size	5
Population Size	50
Mutation Rate	0.015

Personal Instance

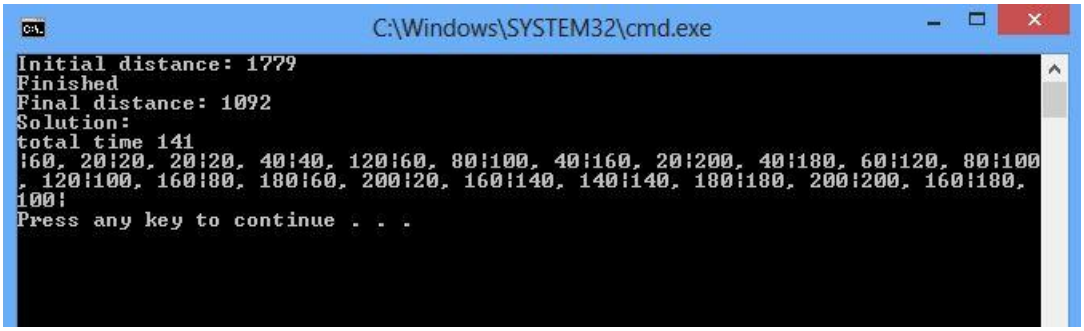

Figure 5. Evolution of GA results over time in Personal Instance

Att-48.tsp

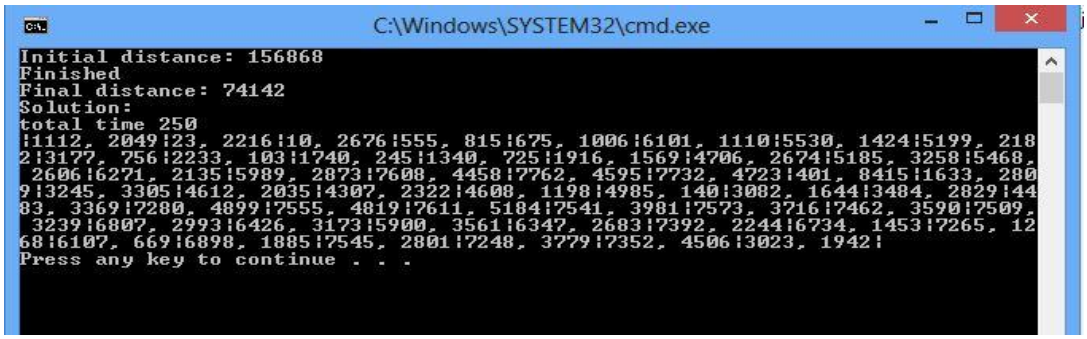

Figure 6. Evolution of GA results over time in Att-48

Berlin52.tsp.

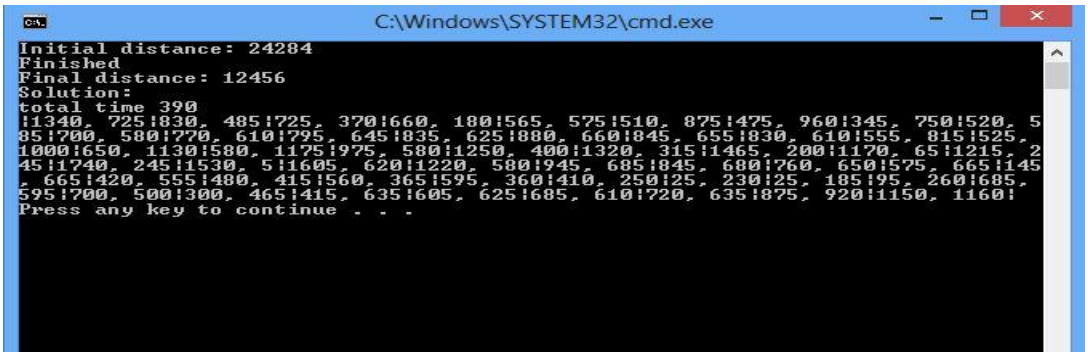

Figure 7. Evolution of GA results over time in Berlin- 52

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

Map Name	Node	Time
Personal instance	20	141
Att-48	48	250
Berlin-52	52	390

(b)Particle Swarm Optimization

Table 3. Basic PSO algorithm parameters values

PSO	
Particle Count	10
Max. Velocity	4
Max. Epochs	10,000

Personal Instance

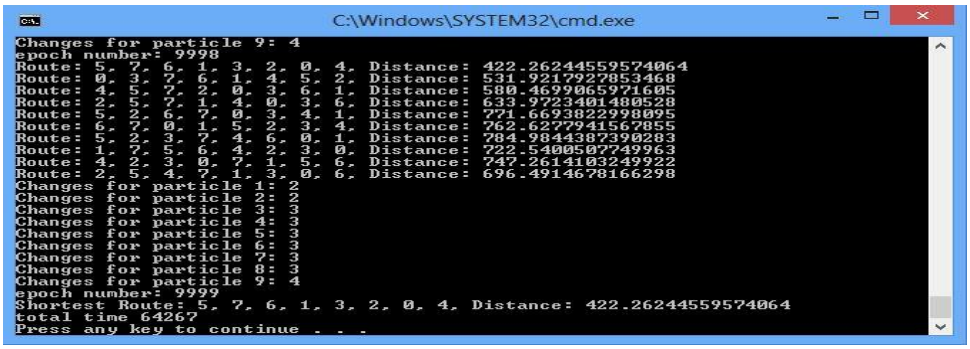

Figure 8. Evolution of PSO results over time in Personal Instance

Att-48.tsp

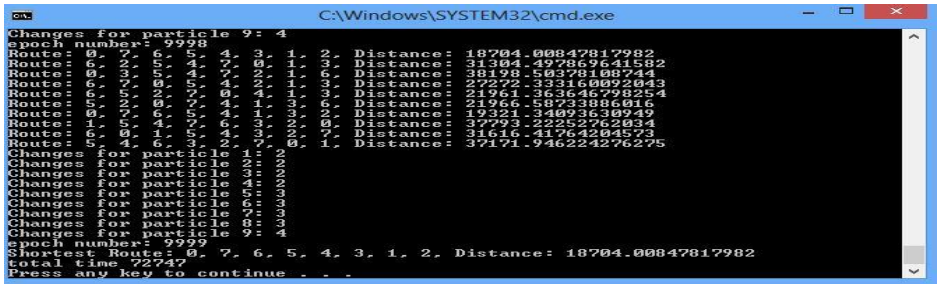

Figure 9. Evolution of PSO results over time in Att-48

Berlin52 .tsp.

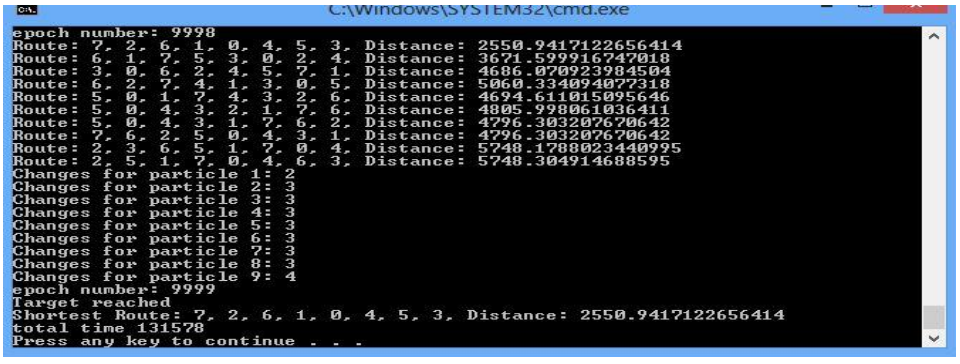

Figure 10. Evolution of PSO results over time in Berlin-52

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

Map Name	Node	Time(in millisecods)
Personal instance	20	64267
Att-48	48	72747
Berlin-52	52	131578

(C) Simulated Annealing (SA)

Table 5. Basic SA algorithm parameters values

Simulated annealing	
Attribute	value
Initial Temp	10,000
Coolong Rate	0.003

Personal Instance

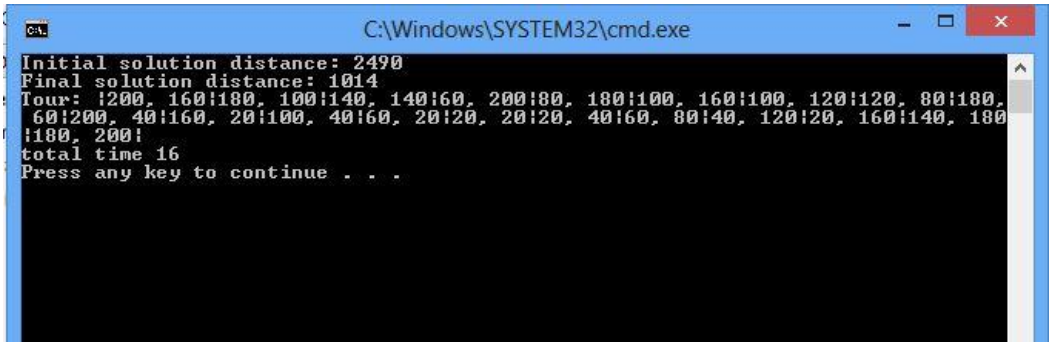

Figure 11. Evolution of SA results over time in Personal Instance

Att-48.tsp

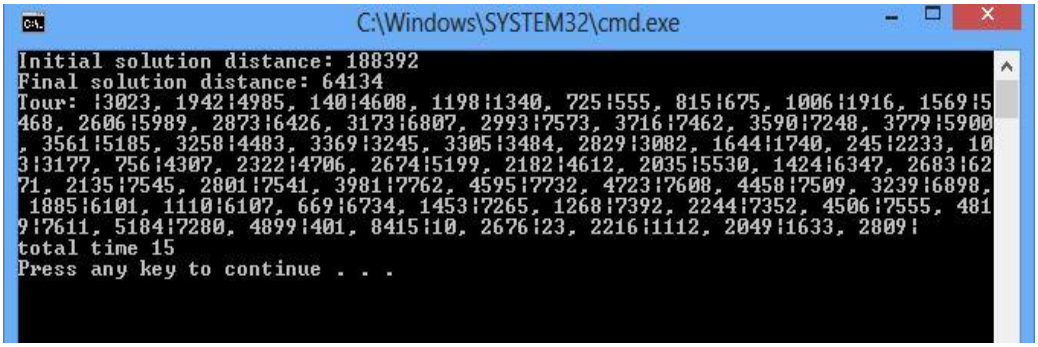

Figure 12. Evolution of SA results over time in Att-48

Berlin52.tsp

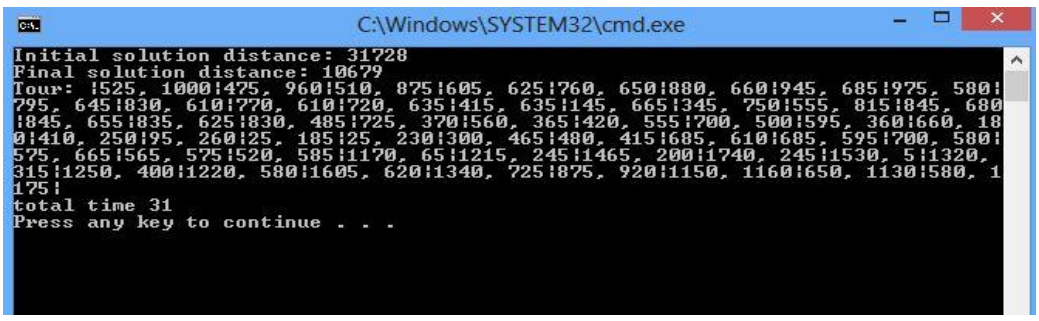

Figure 13. Evolution of SA results over time in Berlin-52

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

Map Name	Node	Time(in millisecods)
Personal instance	20	16
Att-48	48	15
Berlin-52	52	31

Table 7. Overview of Simulation Results

Data Set	GA		PSO		SA	
	Distance (mtr.)	Time (Mil.Sec)	Distance (mtr.)	Time (Mil.Sec)	Distance (mtr.)	Time (Mil.Sec)
Personal	**1092**	**141**	**422**	**64267**	**1014**	**16**
Att-48	**74142**	**250**	**18704**	**72747**	**64134**	**15**
Berlin-52	**12456**	**390**	**2550**	**131578**	**10679**	**31**

Distance Comparison Graph

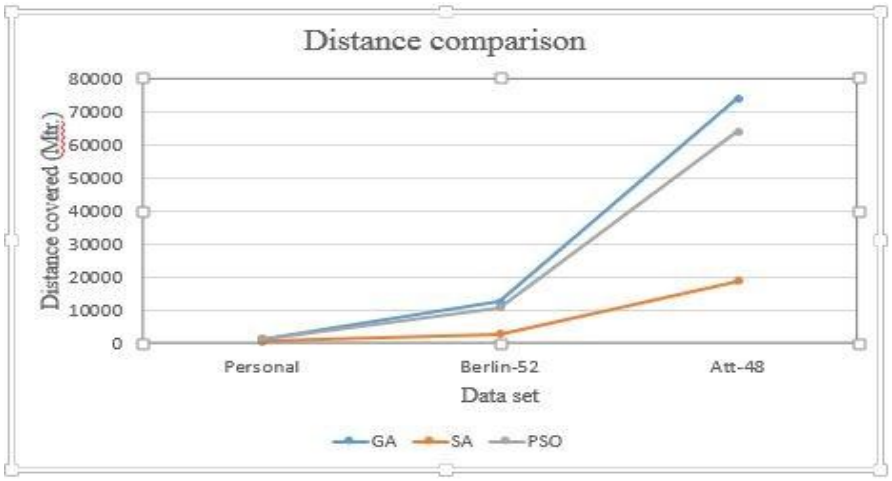

Figure 14. Evolution of GA, PSO and SA results over best distance for Personal Instance, Att-48, and berlin52.

Time Comparison Graph

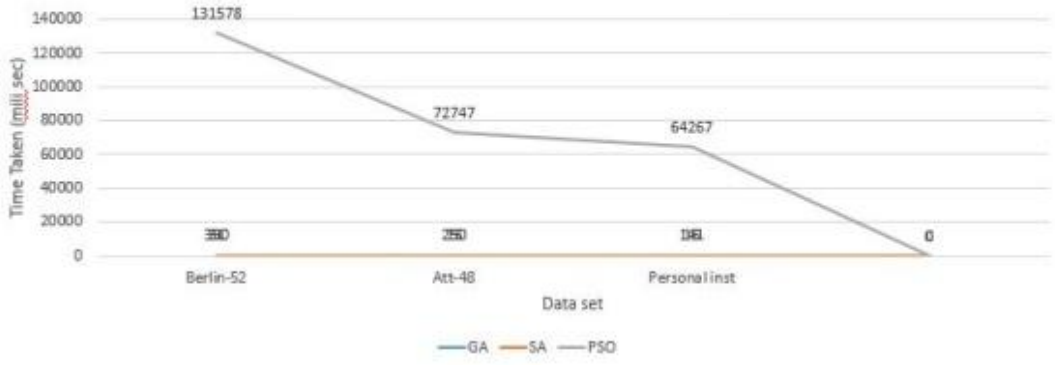

Figure 15. Evolution of GA, PSO and SA results over time(s) for Personal Instance,Att-48, and berlin52**.**

CONCLUSIONAND FUTURE WORK

This paper presented a comparative study view between most widely used optimization algorithm techniques(GA, PSO,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 programming was used to implement the algorithms using two benchmark and one personal instance. We found out that the Simulated Annealing has the shortest execution time 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, PSO stated better than GA and SA. However, PSO appeared in the last order in term of execution time.

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

REFERENCES

[1] S. Arora, Polynomial time approximation schemes for euclidean traveling salesmanand other geometric problems, Journal of the ACM (JACM) 45 (1998) 753–782.

[2]. X. Wang & A. C. Regan, (2002) "Local truckload pickup and delivery with hard time window constraints", Transportation Research, Part B, pp. 97–112.

[3]. M. Shokouhifar& A. Jalali, (2015) "A new evolutionary based application specific routing protocol for clustered wireless sensor networks", AEU-International Journal of Electronics and Communications, Vol. 69, No. 1, pp. 432-441.

[4]. M. Shokouhifar, A. Jalali& H. Torfehnejad, (2015) "Optimal routing in traveling salesman problem using artificial bee colony and simulated annealing", In 1st National Road ITS Congress

[5]. M. Shokouhifar& A. Jalali, (2015) "An evolutionary-based methodology for symbolic simplification of analog circuits using genetic algorithm and simulated annealing", Expert Systems with Applications, Vol. 42, No. 3, pp. 1189-1201.

[6]. C. A. Silvaa, M. C. Sousaa& T. A. Runkler, (2008) "Rescheduling and optimization of logistic processes using GA and ACO", Engineering Applications of Artificial Intelligence, Vol. 21, pp. 343-352.

[7]. C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Computing Surveys (CSUR) 35 (2003)268–308.

[8] F. Glover, G. A. Kochenberger, Handbook of metaheuristics, Springer, 2003.

[9] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Neural Networks, 1995.Proceedings., IEEE International Conference on, volume 4,IEEE, pp. 1942–1948.

[10] D. Teodorovic, P. Lucic, G. Markovic, M. D. Orco, Bee colony optimization: principles and applications, in: Neural Network Applications in Electrical Engineering,2006. NEUREL 2006. 8th Seminar on, IEEE, pp. 151–156.

[11] A. Mucherino, O. Seref, Monkey search: a novel metaheuristic search for global optimization, in: Data Mining, Systems Analysis, and Optimization in Biomedicine(AIPConference Proceedings Volume 953), volume 953, American Institute of Physics, 2Huntington Quadrangle, Suite 1 NO 1, Melville, NY, 11747-4502, USA,, pp. 162–173.

[12] Z. W. Geem, J. H. Kim, et al., A new heuristic optimizationalgorithm: harmonysearch, Simulation 76 (2001) 60–68.

[13] X. S. Yang, Firefly algorithms for multimodal optimization, Stochastic algorithms:foundations and applications (2009) 169–178.

[14] H. Shah-Hosseini, The intelligent water drops algorithm: a nature-inspired swarm based optimization algorithm, International Journal of Bio-Inspired Computation 1(2009) 71–79.

[15] X. S. Yang, S. Deb, Cuckoo search via l´evy flights, in: Nature & BiologicallyInspired Computing, 2009. NaBIC 2009. World Congress on, IEEE, pp. 210–214.

[16]. C. Moon, J. Kim, G. Choi & Y. Seo, (2002) "An efficient genetic algorithm for the traveling salesman problem with precedence constraints," European Journal of Operational Research, Vol. 140, pp. 606-617

[17]. VarshikaDwivedi, TarunaChauhan, SanuSaxena, PrincieAgrawal, "Travelling Salesman Problem using Genetic Algorithm", International Journal of Computer Applications, pp. 25-30 .

[18]. Pang, Wei, et al. "Modified particle swarm optimization based on space transformation for solving traveling salesman problem." *Machine Learning and Cybernetics, 2004.Proceedings of 2004 International Conference on*.Vol.4.IEEE, 2004.

[19]. X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu & Q. X. Wang, (2007) "Particle swarm optimization-based algorithms for TSP and generalized TSP," Information Processing Letters, Vol. 103, pp. 169–176.

[20]. Li, Xiangyong, et al. "A hybrid discrete particle swarm optimization for the traveling salesman problem." *Simulated evolution and learning* (2006): 181-188.

[21]. Allwright, James RA, and D. B. Carpenter. "A distributed implementation of simulated annealing for the travelling salesman problem." *Parallel Computing* 10.3 (1989): 335-338.

[22]. J. H. Holland, "Adaption in Natural and Artificial Systems," The University of Michigan Press, 1975.

[23]. D.E. Goldberg. "Genetic Algorithms in Search, Optimization, and Machine Learning". Addison Wesley, New York, 1989

[24]. Kennedy J, Eberhart R. Particle swarm optimization. IEEE International Conference on Neural Networks. 1995:1942–1948.

[25]. Shi Y, Eberhart R. A modified particle swarm optimizer. IEEE World Congress on Computational Intelligence. 1998: 69–73.

[26]. Kirkpatrick, S., C.D. Gelatt Jr. and M.P. Vecchi, 1983. Optimization by simulated annealing. Science, 220: 671-680.