Volume 4, Issue 2, February 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Comparison on the Performance of Genetic Algorithm and Ant Colony Optimization for Solving the Travelling Salesman Problem

Preet Walia[1]
Department of CSE, BFCET, Bathinda
India

Bharat Naresh Banasal[2]
AP,Department of ECE,MIMT,Malout
India

Harleen Kaur[3]
AP,Department of IT,BFCET,Bathinda
India

Abstract— Evolutionary algorithms are stochastic search methods that mimic the natural biological evolution and/or the social behaviour of species. Examples include how ants find the shortest route to a source of food and how birds find their destination during migration. The behaviour of such species is guided by learning, adaptation, and evolution [1]. To mimic the efficient behaviour of these species, various researchers have developed computational systems that seek fast and robust solutions to complex optimization problems. Such nature inspired algorithms called Heuristic Algorithms which have been developed to arrive at near-optimum solutions and become more popular among researchers to solve real world NP hard problems like Travelling Salesman Problem(TSP), Knapsack Problem, Graph Colouring, Vehicle Routing etc., for which traditional mathematical techniques may fail. In this paper we study on Travelling Salesman problem which is classical optimization problem, that cannot be solved conventionally particularly when no. of cities increase. If one tries to solve TSP using conventional approach by considering all possible tours, it will take years to find optimal solution. So, Heuristic algorithm provide feasible solution to such problem. Here, we consider widely used nature inspired heuristic approaches Ant Colony Optimization, Genetic Algorithm to solve TSP. We also compare results of ACO and GA and tries to find which one gives the better solution in computational time.

Keywords—Travelling Salesman Problem, Ant Colony Optimization, Genetic Algorithm.

I. INTRODUCTION

The difficulties associated with using mathematical optimization on large-scale engineering problems have contributed to the development of alternative solutions. Linear programming and dynamic programming techniques, for example, often fail or reach local optimum in solving NP-hard problems with large number of variables and non-linear objective functions [2]. To overcome these problems, researchers have proposed evolutionary-based algorithms for searching near-optimum solutions to problems.

Evolutionary algorithms are stochastic search methods that used to solve optimization problems like TSP. Travelling Salesman Problem was first formulated in 1930 and is one of the most intensively studied problems in optimization. It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science. If given a set of cities and the distance between each possible pair, the Travelling Salesman Problem is to find the best possible way of 'visiting all the cities exactly once and returning to the starting point'. It is used as a benchmark for many optimization methods. 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 completely and even problems with millions of cities can be approximated within a small fraction of 1%.[3]. In this paper we'll discuss ACO and GA approaches for solving TSP in next subsection.

II. ANT COLONY OPTIMIZATION

The ACO is the oldest and widely used approach proposed by Marco Dorigo in 1992 in his PhD thesis "Optimization, learning, and Natural Algorithms"[4]. The ACO is inspired by the food search behaviour of real ants and their ability in finding the optimum paths. It is a population-based general search technique for the solution of difficult combinatorial optimization problems.

A. Ants' Foraging Behaviour and Optimization

ACO algorithms are based on the mechanics of ants searching for food from a nest area. It was initially suggested for route management problems. Each solution corresponds to an ant's route through the solution space. The objective function value of each solution is computed, and the route taken is noted. Ants that take the same portion of a route influence the acceptability and desirability of the portion of the route. In this manner, the search process evolves.

B. Ant Colony Optimization

This is a population-based, swarm-type heuristic. Ants wander randomly initially, looking for food. Solutions are generated using Monte Carlo simulation. Therefore an ant is a solution. Upon finding the food, ants return to their colony while laying down pheromones. If other ants find a part of the path, they are likely to use it rather than wandering randomly. The path is reinforced if food is found again by another ant. Over time, the path's trail of pheromones evaporates, reducing its attractive strength. The more time it takes for an ant to travel down a trail, the more pheromones evaporate. Shorter paths are travelled more frequently, and pheromones are stronger there as well.

C. The ACO metaheuristic

```
Initialize
SCHEDULE_ACTIVITIES
    • ConstructAntSolutions()
    • DoDaemonActions optional()
    • UpdatePheromones()
END_SCHEDULE_ACTIVITIES
```

These three functions defined as:
ConstructAntSolutions: A colony of ants build a set of solutions. Solutions are evaluated using the objective function.
DaemonActions: Optional centralized actions, e.g.: Local optimization, Ant elitism.
UpdatePheromones: Two opposite mechanisms:(i) Pheromone deposit: Ants increase pheromone values on visited components and/or connections.(ii) Pheromone evaporation: Decrease pheromone trails on all components/connections by a same value.

D. Working of ACO

Moving of Artificial ant depends on the amount of pheromone on the graph edges. The probability of transition of a virtual ant from node i to the node k is given as:

$$p_i^k = \frac{\tau_i^\alpha + \eta_i^\beta}{\sum \tau_{Ni}^\alpha + \eta_{Ni}^\beta}$$

Where, τ_i - indicates the attractiveness of transition in the past, η_i -adds to transition attractiveness for ants, N^i -set of nodes connected to point i, without the last visited point before i, and α, β -parameters found by simulation.

Virtual ant is using same reverse path which saved in its internal memory, but in opposite order and without cycles, which are eliminated. After elimination of cycles, the ant puts pheromone on the edges of reverse path according to formula

$$\tau_{ij}^{t+1} = \tau_{ij}^t + \Delta\tau^t$$

Where, τ_{ij}^t -value of pheromone in step t, $\Delta\tau$ -value by ants saved pheromones in step t, which will be constant or can be changed depends upon solution equality.

At last, pheromones on the edges evaporated. This helps to find shortest path. This evaporation of pheromones has an intensity ρ .

$$\tau_{ij}^{t+1} = (1-\rho)\tau_{ij}^t$$

Where, This formula applied on the graph edges with intensity ρ (interval(0,1)).

E. Advantages of ACO

- Distributed computing and Robust.
- Easy to accommodate with other algorithms.
- Its major advantage over Genetic Algorithm and Simulated Annealing of similar problems(like TSP) when graph changes dynamically, the ant colony algorithms can be run continuously and adapt to changes in real time[5].

F. Disadvantages of ACO

- Though ant colony algorithms can solve some optimization problems successfully, we cannot prove its convergence. [6]
- It is prone to falling in the local optimal solution. because the ACO updates the pheromone according to the current best path.[7]

III. GENETIC ALGORITHMS

Pioneered by John Holland in the 1970's. Got popular in the late 1980's. Based on ideas from Darwinian Evolution. Can be used to solve a variety of problems that are not easy to solve using other techniques. Genetic algorithms are implemented as a computer simulation in which a population of abstract representations (called chromosomes or the genotype or the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible.

A. Inspiration

Inspired by natural evolution. Population of individuals where individual is feasible solution to problem. Each individual is characterized by a Fitness function. Higher fitness is better solution. Based on their fitness, parents are selected to reproduce offspring for a new generation. Fitter individuals have more chance to reproduce. New generation has same size as old generation; old generation dies. Offspring has combination of properties of two parents. If well designed, population will converge to optimal solution.

- An individual is characterized by a set of parameters: Genes
- The genes are joined into a string: Chromosome
- The chromosome forms the genotype
- The genotype contains all information to construct an organism: the phenotype
- Reproduction is a "dumb" process on the chromosome of the genotype.

B. GA Operators and Termiologies

- Encoding: the process of representing solution in form of string that conveys the necessary information. Just as in chromosome, each gene controls a particular characteristics of the individual, similarly each bit in the string represents a characteristics of the solution.
- Fitness Evaluation: fitness function quantifies optimality of solution(chromosome). So that particular solution may be ranked against all other solutions. A fitness value is assigned to each solution depending on how close it actually is to solving the problem.
- Reproduction: it is the process of getting new offspring for the mating pool. Two types which are crossover and mutation.
- Survivor Selection: selection is the process of choosing one or two parents from the population for reproduction. The higher fitness value the more chance to be selected, which will increase the convergence rate.

C. Genetic Algorithm

```
BEGIN
        Generate initial population;
        Compute fitness of each individual;
        REPEAT /* new generation*/
                FOR population_size/2 DO
                Select two parents from old generation;
                /*biased to fitter ones*/
                Recombine parents for two offsprings;
                Compute fitness of offsprings;
                Insert offsprings in new generation;
        END FOR
UNTIL population has converged
END
```

D. Working of GA

GA uses three operations to maintain population that evolve from one generation to another.

- "Selection" operation which is inspired by the principle of 'Survival of the Fittest'. The search begins from a randomly generated population that evolve over successive generations (iterations).A fitness function is used to evaluate the performance of the solutions. Each time two solutions are chosen as parent solutions by selection process based on fitness function. Chance to be selected as parent proportional to fitness
  - Roulette wheel

  To avoid problems with fitness function
  - Tournament
- "Crossover" operation, which is inspired by mating in biological populations. The crossover operator inherits features of good surviving designs from the parent population into the future population, which will have better fitness value on average. Generating offspring from two selected parents:
  - Single point crossover
  - Two point crossover (Multi point crossover)
  - Uniform crossover
- "Mutation" which causes diversity in population characteristics. It causes local modifications to the new generation randomly. The new generation is identical to the parent except one or more changes made by mutation process. Crossover can only explore the combinations of the current gene pool. Mutation can "generate" new genes.
- Repeat selection, crossover and mutation operations to produce more new solutions until the population size of the new generation is the same as that of the old one.
- The iteration then starts from the new population. Since better solutions have a larger probability to be selected for crossover and the new solutions produced carry the features of their parents, it is hoped that the new generation will be better than the old one. The procedure continues until the number of generations is reached to n or the solution quality cannot be easily improved.

E. Advantages of GA

- Parallelism and solution space is wider.
- Easy to discover global optimum.
- Only uses function evaluation and handles noisy functions well[8].
- Handles large, poorly understood search spaces easily.
- They require no knowledge or gradient information about response surface.

- Discontinuities in response surface slightly effect the solution.

F. Disadvantages of GA

- The problem of identifying fitness function.
- The problem of choosing various parameters like size of population, mutation rate , crossover, the selection method and its strength.
- No good at identifying local optima and not also effective terminator.

IV. EXPERIMENT RESULTS

By using ACO and GA, we solve the Travelling salesman Problem having different cities. The code used to execute in MATLAB, window 8 platform with Intel core i5 processor and 3GB RAM. In Table 1, there is defined ACO algorithm for solving TSP and 10 different graphs which showing the paths. In Table 2, defined GA for solving TSP with same no. of cities respectively and also graphs

Table 1. ACO algorithm used to solve TSP

Cities	**Time**	**Distance**	**Feasible solutions**
4	1.40 sec	14.28	4-1-2-3
5	1.50 sec	34	1-5-2-4-3
7	1.78 sec	114.3	7-1-5-3-4-2-6
11	2.72 sec	140	4-2-9-8-1-11-5-10-3-6-7
12	3.13 sec	1733	9-7-11-5-3-10-1-6-8-12-4-2
15	5.14 sec	291	1-13-2-15-9-5-7-3-12-14-10-8-6-4-11
22	10.4 sec	873	9-13-19-15-17-14-2-16-8-11-21-7-4-3-10-20-22-5-12-1-18-6
30	19.4 sec	527	18-3-22-13-24-11-6-2-4-28-30-9-14-15-29-10-25-27-12-8-20-17-19-16-21-26-5-23-7-1
57	1.30min	14168	30-22-10-12-42-1-11-7-6-36-39-19-3-56-44-52-24-2-4-34-17-40-29-47-43-48-49-20-46-14-8-5-41-33-15-37-26-54-57-55-13-21-35-23-31-28-51-25-45-50-38-9-32-18-53-16-27
128	9.51min	23546	38-22-73-34-54-18-81-127-128-66-126-107-103-78-12-35-74-60-110-88-98-39-9-99-8-41-67-63-83-26-29-121-56-19-27-17-53-96-57-97-118-15-16-36-106-84-37-13-59-49-58-82-45-112-30-90-4-93-120-6-94-5-91-1-72-70-48-71-69-40-61-117-105-114-108-47-102-123-62-46-20-111-51-100-85-32-2-76-33-124-64-65-95-116-10-79-24-109-89-3-50-28-43-80-21-52-55-119-113-68-7-122-25-23-77-44-92-42-87-11-31-86-101-14-31-86-101-14-115-75-125-104

ACO Graphs with different cities

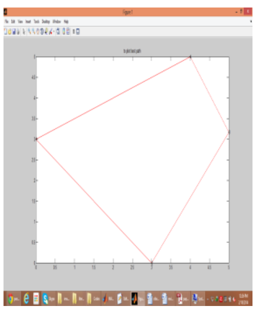
Fig 1. 4 cities

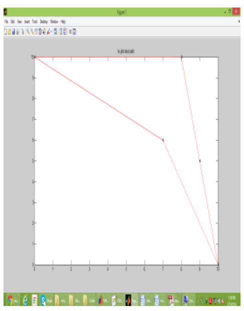
Fig 2. 5 cities

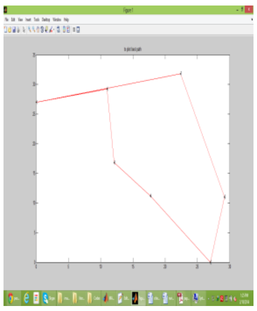
Fig 3. 7 cities

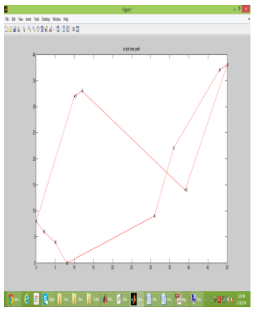
Fig 4. 11 cities

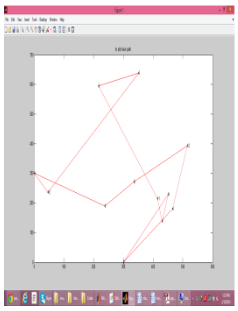
Fig 5. 12 cities

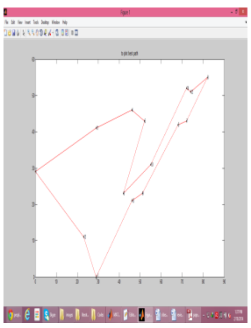
Fig 6. 15 cities

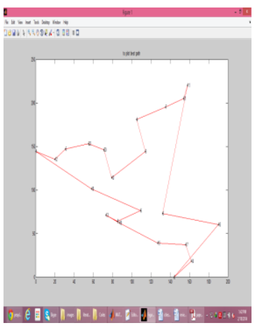
Fig 7. 22 cities

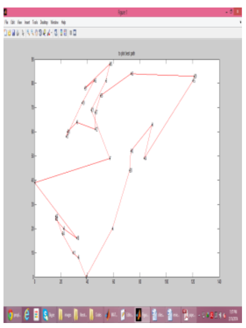
Fig 8. 30 cities

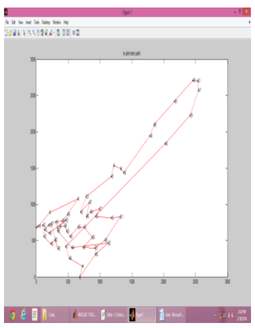
Fig 9. 57 cities

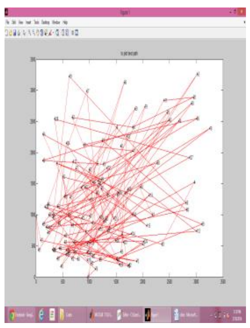
Fig 10. 128 cities

Table 2. GA used to solve TSP

Cities	Time	Distance	Feasible solutions
4	1.40 sec	21.966	3-4-1-2
5	1.50 sec	51.9619	1-5-2-4-3
7	1.80 sec	220.7286	2-6-7-1-5-3-4
11	1.50 sec	553.0665	9-11-1-5-8-7-10-3-6-4-2
12	2.19 sec	7393.9912	8-2-9-12-4-3-5-1-6-10-11-7
15	2.22 sec	1253.4645	12-10-6-14-9-5-2-8-13-15-1-4-11-7-3
22	2.93 sec	5299.2571	22-1-12-5-13-6-19-9-8-10-4-7-20-17-21-11-16-14-18-2-15-3
30	3.34 sec	4740.3948	26-23-2-30-17-12-18-6-22-1-5-3-7-10-4-29-15-24-19-27-14-9-28-16-21-25-11-13-8-20
57	5.16 sec	254998.3714	6-7-47-49-43-32-40-50-39-24-44-52-21-38-48-14-41-19-17-55-33-20-46-22-8-4-28-29-36-54-1-18-2-34-3-56-37-27-23-11-9-13-57-1-16-30-25-15-53-10-31-12-42-5-35- 45-26
128	10.24sec	1509811.5141	96-3-31-65-45-471-101-53-116-2-36-18-19-105-24-43-20-70-126-125-78-95-58-60-69-107-25-77-14-34-61-17-54-92-108-93-87-57-7-9-33-99-91-40-103-59-39-50-12-71-35-81-72-84-97-56-4-13-111-79-5-74-11-100-67-115-66-80-6-64-30-26-15-75-32-121-49-29-117-120-41-104-76-38-106-85-124-27-8-123-51-23-127-83-88-44-37-16-10-122-1-21-42-112-63-109-114-110-22-48-82-28-94-90-119-62-52-113-128-118-73-46-89-98-68-102-86-55

Genetic Algorithm Graphs with different cities

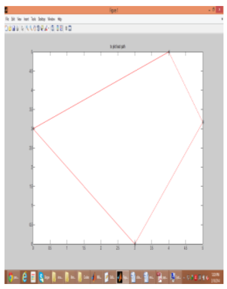
Fig 11. 4 cities

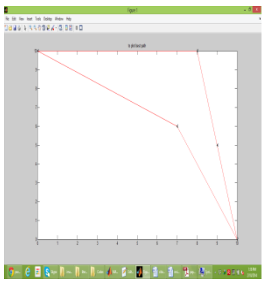
Fig 12. 5 cities

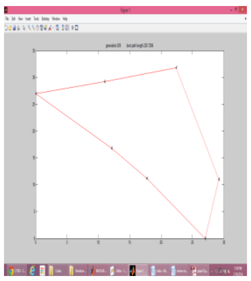
Fig 13. 7 cities

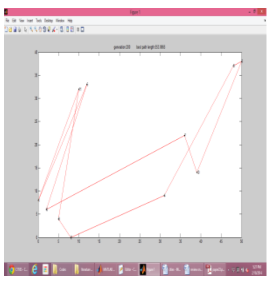
Fig 14. 11 cities

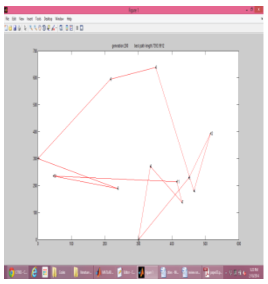
Fig 15. 12 cities

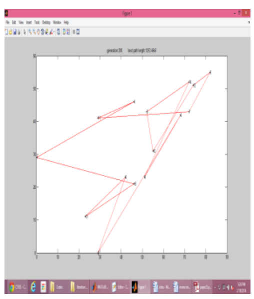
Fig 16. 15 cities

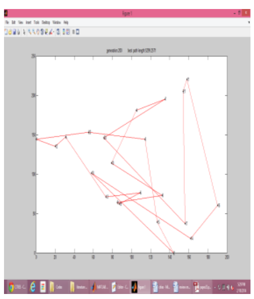
Fig 17. 22 cities

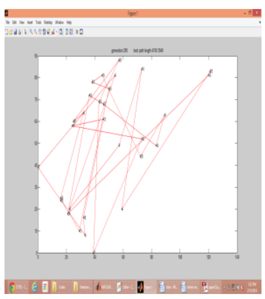
Fig 18. 30 cities

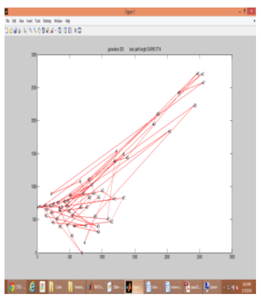
Fig 19. 57 cities

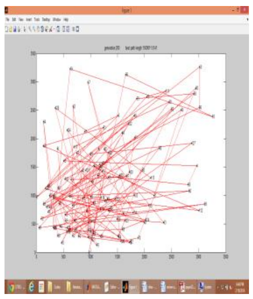
Fig 20. 128 cities

V. CONCLUSION

In this paper, we have presented the comparative analysis of Ant Colony Optimization and Genetic Algorithm for Travelling Salesman Problem. The proposed work shows that the ACO gives better results as compare to the GA. ACO graphs show simple paths or feasible solutions also when cities increased in number and also show less distance. On other side, GA shows simple path only when cities less in number but when cities increased then GA shows complex paths and also high distances.

REFERENCES

[1] Applegate, D. L.; Bixby, R. M.; Chvátal, V.; Cook, W. J. (2006), The Traveling Salesman Problem, ISBN 0-691-12993-2.

[2] Lovbjerg M. Improving particle swarm optimization by hybridization of stochastic search heuristics and self-organized criticality. Masters Thesis, Aarhus Universitet, Denmark; 2002.

[3] Applegate, D. L.; Bixby, R. M.; Chvátal, V.; Cook, W. J. (2006), The Traveling Salesman Problem, ISBN 0-691-12993-2.

[4] M. Dorigo, L. Gambardella, "Ant colonies for the Traveling salesman problem." Biosystems 43, pp. 73-81, 1997.

[5] Shwetasinghal, shivangigoyal, shubhragoyal and divyabhatt, "A comparative study of a class of Nature Inspired Algorithms", Proceedings of the 5th national conference :INDIACom, March 10-11, 2011.

[6] X. Yuneig, G. Junen, G.Bo, "An Ant Colony Optimization Algorithm based on the Nearest Neighbour Node Choosing Rules and the Crossover Operator", International Conference on Computer Science and Software Engineering, 2008.

[7] Peiyi Zhao, Peixin Zhao, Xin Zhang, "A New Ant Colony Optimization for the Knapsack Problem", 2007.

[8] Mithun Kuniyuil ,"Introduction to Genetic Algorithms", 2010.