Ant Colony Optimization Meta-heuristic for Solving Real Travelling Salesman Problem

Sourabh Joshi and Sarabjit Kaur

Abstract Ant colony optimisation is a population-based advanced approach for finding the solution of difficult problems with the help of a bioinspired approach from the behaviour of natural ants. The ant colony algorithm is a propelled optimisation method which is utilised to take care of combinatorial optimisation problems. The significant features of this algorithm are the utilisation of a mixture of preinformation and postinformation for organizing great solutions. The ant colony algorithm is used in this paper for solving the travelling salesman problem of the real set of data and getting the optimal results on graphs. This algorithm is an meta-heuristic algorithm in which we used the 2-opt local search method for tour construction and roulette wheel selection method for selection of nodes while constructing the route. The results show that this algorithm can efficiently find the optimal path of the hundred cities with minimum time and cost.

Keywords Ant colony algorithm · Metaheuristic · Genetic algorithm · Travelling salesman problem

1 Introduction

Ant colony optimisation (ACO) is a bioinspired mechanism used in genetic programming to solve complex probabilistic problems. ACO was invented in 1992 by Marco Dorigo in his PhD thesis [1]. The ant colony optimisation algorithm is widely used because of the use of positive feedback mechanism, heuristic probability, computing distributed numeric information, and other characteristics. The ACO algorithms are used for solving the different problems of discrete mathematics and operation search such as the quadratic assignment problem and travelling salesman problem (TSP). In the ACO, we use a mixture of some advance

Sourabh Joshi (✉) · Sarabjit Kaur
Department of CSE, CT Institute of Technology & Research,
Jalandhar 144002, India
e-mail: er.sourabhjoshi@gmail.com

N.R. Shetty et al. (eds.), *Emerging Research in Computing, Information, Communication and Applications*, DOI 10.1007/978-981-10-0287-8_5

and run-time knowledge for making decisions in the formula. TSP is a NP-hard problem which can be developed to be an admissible solution for any other problem that belongs to the NP-hard class [2]. In this paper roulette wheel selection is used, which is the most democratic selection method where the selection of parents is based upon their fitness. The better chromosomes are selected so that they can give feasible results. The algorithm generates random numbers in the initial phase of tour construction to select the city randomly based on probability of finding the city in the search area [3]. The other sections of this paper are as follows; Sect. 2 gives the background detail of TSP and explains the algorithm. Section 3 explains the roulette selection which is used in the ACO framework and Sect. 4 describes the implementation part with the test and results.

2 Ant Colony Optimisation for TSP

2.1 TSP

The problem of finding an optimal path between n number cities is known as TSP. The TSP is a most significant problem first posed by the Irish mathematician W. R. Hamilton in the nineteenth century [4]. This problem has also been intensely studied in operations research and other areas since 1930. Formally, TSP can be represented by a complete weighted graph, $G = (V, E)$, in which V represents a set of n vertices and E represents a set of bidirectional edges between V_i, $V_j \in V$, and minimize the vertices $\sum n_i = 0\ W_{i,j}$, where $W_{i,j}$ is used to represent an edge weight between two vertices V_i and V_j.

2.2 Ant Colony Optimisation

The ant colony meta-heuristic is an advanced approach for finding the solution of difficult problems with the help of a bioinspired approach from the behaviour of natural ants [5]. In ACO we use artificial ants which work similarly to natural ants for searching a good solution of the optimisation problem, whereas applying the ACO optimisation problem converts it into a weighted graph problem for finding the best feasible path [6]. The ACO algorithm is divided into three main parts as follows.

Tour Construction We start constructing our tour by placing ants at random places (vertices) in our graph where each ant will decide which is the best path to reach the next vertex by taking the next move based on the formula

$$\rho_{xy}^{k} = \begin{cases} \frac{\left[\tau_{xy}(t)\right]^{\alpha}\left[\eta_{xy}\right]^{\beta}}{\sum_{\mu \in J_{k(x)}}\left[\tau_{x\mu}(t)\right]^{\alpha}\left[\eta_{x\mu}\right]^{\beta}} & \text{if } y \in J_{k(x)} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where ρ is the probability of the ant to move from one vertex to another. The variable τ is used to represent the quantity of pheromone while searching for food, and α is a heuristic constant which is used for finding the paths due to its greedy approach [7]. This is the case where the inverted distance is 1/distance between the city x and y and raised to the power of β is also a heuristic constant which describes the speed of selecting paths by ants and everything calculated until it is divided by the summations of every solution. The record of cities where the ant k passes is kept in the tabu list (tabuk) [8].

Pheromone Update After construction of the tour, updating of residual information is performed when all the ants finish their traversing. This is the formula:

$$\tau_{xy}(t+n) = (1-\rho) \times \tau_{xy}(t) + \Delta\tau_{xy}(t) \quad (2)$$

where

$$\Delta\tau_{xy}(t) = \sum_{k=1} m\Delta\tau_{xy}^{k}(t) \quad (3)$$

τ is the absolute pheromone amount which gets deposited for worker (ant) k on the edge xy. ρ refers to the pheromone volatisation coefficient and $(1 - \rho)$ represents the delay of pheromone ranges 0–1 [9].

Terminating Condition If the terminating condition is satisfied (i.e., all the cities are visited and no city is repeated), the circulation will stop. Compare all the best solutions previously updated in the tabu list (tabuk) in every iteration and find the optimal solution; otherwise empty the tabu list and continue the iteration.

2.3 Genetic Algorithm

A genetic algorithm approach is inspired from the chromosomes which are used to solve complex problems [10]. They are used to handle the population of possible solutions where each solution represents a chromosome containing an abstract representation [11]. The genetic algorithm iteration has these phases:

Selection phase The selection process defines the fitness of randomly selected individuals.
Reproduction This process uses both recombination and mutation for producing new chromosomes.

Evaluation In this process evaluation is done on the basis of fitness of new chromosomes.
Replacement In this process, old chromosomes are replaced by the new ones.

3 Roulette Wheel Selection

In the roulette wheel selection method [12], we use the random number generation mechanism for selecting the best path. In the roulette wheel selection mechanism the spinning of each segment is according to the probabilities of selecting parent values with the most fitness having more probability to be chosen. The largest segment is occupied by the fittest individual whereas in correspondence the smaller segment is occupied by the least fit within the roulette wheel [13]. The circumference of the roulette wheel is considered as the sum of all segments on the surface of the wheel [14]. In this selection mechanism, selection is done by finding the probability of individual K, P(Choice zK) as defined with the equation as

$$P(\text{Choice zK}) = \frac{\text{def fitness } (K)}{\sum n_{jz}, \text{fitness } (j)} \tag{4}$$

Here's some pseudo-code of the roulette wheel selection for computing fitness of the individual.

```
Algorithm: Roulette wheel Selection ()
r :=random number ,0 ≤r ≤ 1 ;
Sum: 0;
For each individual K
{
        Sum: = sum +P (ChoicezK)
        If r <Sum;
        return;
}
```

4 Experiments and Results

4.1 Problem Formulation

The aim of this paper is to solve the TSP problem which is a NP-hard problem by using the ACO meta-heuristic. NP problems are nondeterministic polynomial time problems; the NP-complete problems are hard once whose solutions can deal with any other NP problem in polynomial time. We can also find suboptimal solutions of NP problems which may be found in polynomial time.

Table 1 100 cities on which optimisation is performed

S. no.	Cities	S. no.	Cities
1.	Saharanpur–Shamli	51.	Varanasi–Chanauli
2.	Shamli–Muzaffar Nagar	52.	Chandauli–Sasaram
3.	Muzaffar Nagar–Bijnor	53.	Sasasram–Aurangabad
4.	Bijnor–Meerut	54.	Aurangabad–Gava
5.	Meerut–Baghpat	55.	Gava–Nawada
6.	Baghpat–Gaziabad	56.	Nawada–Shiekhpura
7.	Gaziabad–Noida	57.	Shiekhpura–Lakhisarai
8.	Noida–Hapur	58.	Lakhisarai–Munger
9.	Hapur–Bulandshahar	59.	Munger–Khagaria
10.	Bulandshahar–Aligarh	60.	Khagaria–Katihar
11.	Aligarh–Hathras	61.	Katihar–Purnia
12.	Hatras–Mathura	62.	Purnia–Araria
13.	Mathura–Agra	63.	Araria–Supaul
14.	Agra–Firozabad	64.	Supaul–Madhepura
15.	Firozabad–Etah	65.	Madhepura–Saharsa
16.	Etah–Mainpuri	66.	Saharsa–Samastipur
17.	Mainpuri–Etawah	67.	Samastipur–Darbhanga
18.	Etawah–Auraiya	68.	Darbhanga–Madhubani
19.	Auraiya–Akbarpur	69.	Madhubani–Sitamarhi
20.	Akbarpur–Orai	70.	Sitamarhi–Muzzafarpur
21.	Orai–Jhansi	71.	Muzzafarpur–Hajipur
22.	Jhansi–Lalitpur	72.	Hajipur–Chhapra
23.	Lalitpur–Datia	73.	Chhapra–Siwan
24.	Datia–Shivpuri	74.	Siwan–Gopalganj
25.	Shivpuri–Badarwas	75.	Gopalganj–Gorakhpur
26.	Badarwas–Guna	76.	Gorakhpur–Khalilabad
27.	Guna–Rajgarh	77.	Khalilabad–Basti
28.	Rajgarh–Shajapur	78.	Basti–Faizabad
29.	Shajapur–Ujjain	79.	Faizabad–Bora Banki
30.	Ujjain–Dewas	80.	Bora Banki–Lukhnow
31.	Dewas–Indore	81.	Lukhnow–Hardoi
32.	Indore–Dhar	82.	Hardoi–Sitapur
33.	Dhar–Jhabua	83.	Sitapur–Shahjahanpur
34.	Jhabua–Alirajpur	84.	Shahjahanpur–Kashganj
35.	Alirajpur–Barwani	85.	Kashganj–Budaun
36.	Barwani–Khargane	86.	Budaun–Bareilly
37.	Khargane–Khandwa	87.	Bareilly–Pilibhit
38.	Khandwa–Harda	88.	Pilibhit–Rampur
39.	Harda–Betul	89.	Rampur–Rudrapur
40.	Betul–Chhindwara	90.	Rudrapur–Nainital

(continued)

Table 1 (continued)

S. no.	Cities	S. no.	Cities
41.	Chhindwara–Seoni	91.	Nainital–Almora
42.	Seoni–Balaghat	92.	Almora–Bageshwar
43.	Balaghat–Mandla	93.	Bageshwar–Chamoli
44.	Mandla–Dindori	94.	Chamoli–Rudraprayag
45.	Dindori–Shahdol	95.	Rudraprayag–Pauri
46.	Shahdol–Umaria	96.	Pauri–Dehradun
47.	Umaria–Katni	97.	Dehradun–Ambala
48.	Katni–Rewa	98.	ambala–yamunanagar
49.	Rewa–Mirzapur	99.	Yamunanagar–Sirmaur
50.	Mirzapur–Varanasi	100.	Sirmaur–Saharnpur

4.2 Experimental Setup

While going for the implementation part, we solve the NP problem taking an example of a travelling salesman problem in which a travelling salesman wants to visit 100 different cities by driving through roads, starting and ending his trip at home. This algorithm is performed in MATLAB®. The results show the graphical output of 100 cities of different states in India and generate the optimal solution with respect to iterative time and iterative best cost.

4.3 Experimental Results

While performing the experiment to solve TSP we have to define the starting and ending nodes of our round trip so that we can easily calculate the distance of the round trip. In this real-life example we collect the data of 100 cities which a travelling salesman has to visit by defining "Saharanpur" as the starting and ending node of our round trip as shown in Table 1. In the table we describe the number of cities [15] and their respective names [16]. As defined in our problem the salesperson has to visit 100 cities for delivering products following a shortest route where the salesperson has to visit each city exactly once. In our results the salesperson visits all 100 cities defined in the table with the optimal distance of 6435.302 km.

In the ant colony algorithm we take these parameters' values as $\alpha = 1$, $\beta = 5$, $Q = 10$, $C = 100$, $\rho = 0.65$, and $\lambda = 0.15$. In this TSP problem we take 100 different locations with their geocoordinate values in the input file and set input type as geo. Figure 1 shows the output path of the optimised route which is then drawn on the map of India. Figure 2 shows the states it covers and Fig. 3 shows the graph of the number of iterations with respect to the cost of the optimal path calculated.

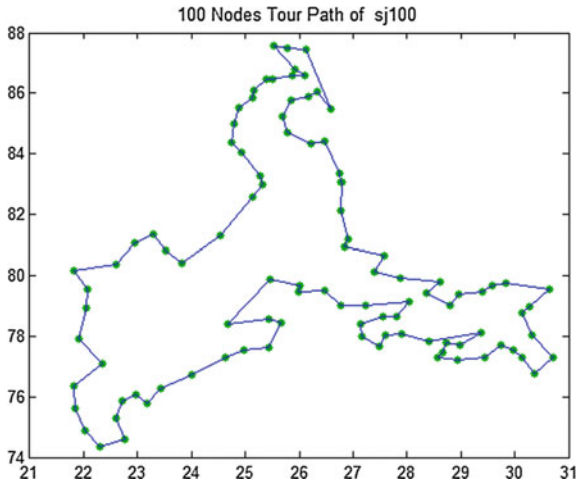

Fig. 1 The output path of the optimised route which is then drawn on the map of India

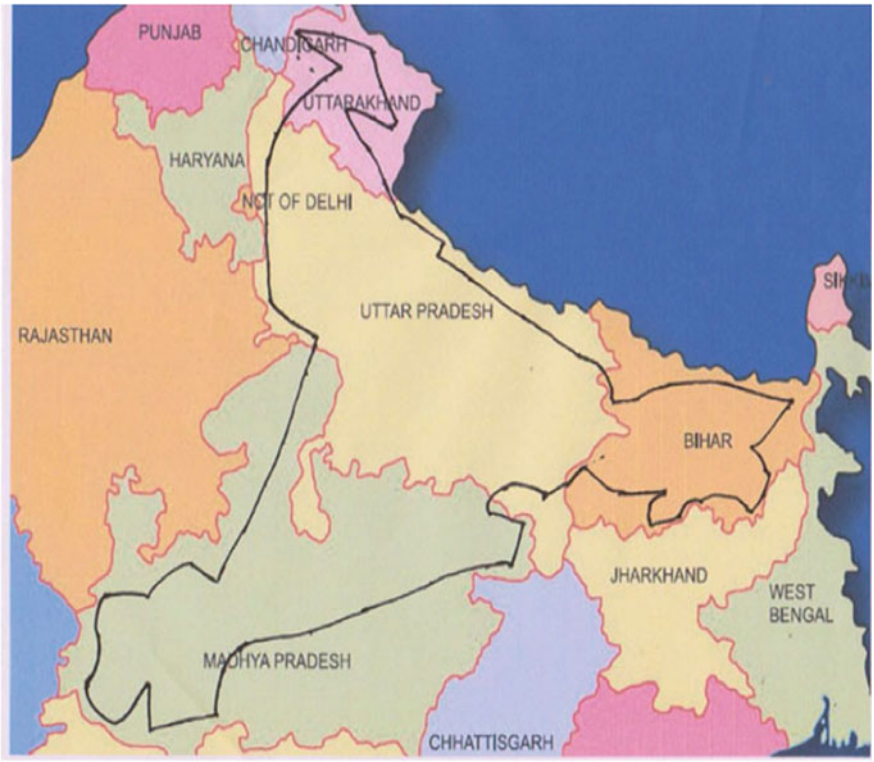

Fig. 2 The states it covers

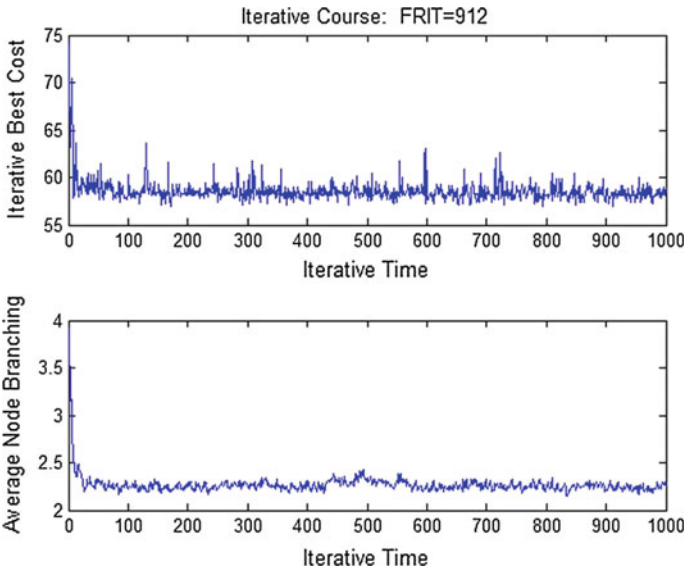

Fig. 3 The graph of the number of iterations with respect to the cost of the optimal path calculated

5 Conclusion

We concluded from the proposed work that the roulette wheel selection method is an efficient selection method which is used in the ant colony optimisation (ACO) algorithm for finding the optimal route of the travelling salesman problem. The results also revealed that the roulette-based selection explores the search space and visits the 100 cities with the lowest iterative cost with respect to iterative time for the defined tour. Future work could be evaluated by using the variable neighbourhood search heuristic in the ACO framework with the combination of different selection strategies such as the tournament-based selection strategy and rank-based selection strategy for the travelling salesman problem and its variants.

References

1. Escario, J.B., Jimenez, J.F., Giron-sierra, J.M.: Ant Colony Extended: Experiments on the Travelling Salesman Problem (2014)
2. Adham, M.T., Bentley, P.J.: An artificial ecosystem algorithm applied to the travelling salesman problem. In: Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion—GECCO Comp '14, pp. 155–156 (2014)
3. Wei, X., Han, L., Hong, L.: A modified ant colony algorithm for traveling salesman problem. IJCCC **9**(5), 633–643 (2014)
4. Yu, Y., Chen, Y., Li, T.: A new design of genetic algorithm for solving TSP. Comput. Sci. Optim. (CSO) (2011)

5. O'Neill, M., Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming. In: Genetic Programming and Evolvable Machines, Mar 2008
6. Science, C., Engineering, S.: An approach to combinatorial problems by mapreduce based ant colony optimization. **4**(1), 1009–1014 (2014)
7. Jing, S., Yan-ping, B., Hong-ping, H., Jin-na, L.: Using the Improved Ant Colony Algorithm to Solve the Chinese TSP. In: 2014 International Conference on Future Computer and Communication Engineering (ICFCCE 2014) (2014)
8. Dorigo, M., Stutzle, T.: Ant Colony Optimization: Overview and Recent Advances‖, Technical Report No. TR/IRIDIA/2009-013, pp. 1–32 (2009)
9. Runka, A.: Evolving an Edge Selection Formula for Ant Colony Optimization. 08 July 2009
10. Xu, S., Wang, Y., Huang, A.: Application of imperialist competitive algorithm on solving the traveling salesman problem. Algorithms **7**, 229–242 (2014)
11. Mavrovouniotis, M.: Ant colony optimization with self-adaptive evaporation rate in dynamic environments. no. CCI. In: Oliver, R., Rickard, N. (eds.) Efficiently Vectorized Code for Population Based optimization Algorithms. 28 Mar 2013
12. Wei, X.: Parameters analysis for basic ant colony optimization algorithm in TSP. **7**(4), 159–170 (2014)
13. Meşecan, İ., Bucak, İ.Ö., Asilkan, Ö.: Searching for the shortest path through group processing for TSP. Math. Comput. Appl. **16**, 53–65 (2011)
14. Noraini, M.R., Geragthy, J.: Genetic Algorithm Performance With Different Selection Strategies in Solving TSP, (WCE 2011). London, U.K
15. Kanoh, H., Ochiai, J., Kameda, Y.: Pheromone trail initialization with local optimal solutions in ant colony optimization. Int. J. Knowl.-Based Intell. Eng. Syst. **18**, 11–21 (2014)
16. Orld, R.E.A.L., Roblems, D.E.P., Rea, W.I.D.E.A., Etwork, R.O.A.D.N.: Hybrid Ant Colony Optimization for on Real Time and Predicted Traffic In, pp. 379–389 (2014)