Recent Trends in Communication and Computer Networks , ComNet

# EOT: Evolutionary Optimization Technique for Solving Traveling Salesman Problem

[1] Sana Abdul Rahman Kutty and [2]Sindhya K Nambiar
[1-2] Department of Computer Science & Engineering,
SCMS School of Engineering & Technology, Ernakulam, Kerala, India
Email: sana.arkutty@gmail.com, sindhyak@yahoo.com

*Abstract*—**Various combinatorial optimization problems like Traveling Salesman Problem are solved using numerous nature inspired optimization techniques. In this paper, Magnetic Opti- mization Algorithm and Ant Colony Optimization techniques are described for solving Traveling Salesman Problem. In magnetic optimization algorithm, each magnetic agent or particle in MOA is represented by a candidate solution of Traveling Salesman Problem. The magnetic force between the particles strength is inversely proportion to distance calculated by TSP. Particles with higher magnetic force, attract relatively lower magnetic force particles. The process is repeated until a stopping condition is satisfied, and the best found solution will be considered the solution with lowest distance. The Ant Colony Optimization (ACO) is an evolutionary techniques to solve TSP. The quality of the solution is further compared with the optimal solution and computation time.**

*Index Terms*— **Traveling Salesman Problem(TSP), Magnetic Optimization Algorithm(MOA), Ant Colony Optimization(ACO), combinatorial optimization.**

## I. INTRODUCTION

Traveling Salesman Problem, a well-known problem which includes the study of finding the shortest probable distance that a salesman has to travel by visiting all the cities only once and return back to the starting city. Application of TSP is found in many areas such as in very large scale integrated circuit (VLSI) for wires routing and routing for automated holes drilling process. Additional distance covered involve additional costs in term of cost or time. This is a great motivation to propose new techniques in solving TSP effectively.
Magnetic Optimization Algorithm (MOA) is a novel heuris- tic optimization algorithm were the inspiration is from the principles of magnetic field theory. This algorithm has a good performance for solving optimization problems as in [6]. The search is carried out as search agents by using magnetic particles whereas interaction between particles with each other is based on law of electromagnetic force, in this algorithm. MOA has designed to solve problems in continuous real search spaces (domains). Here TSP is solved using MOA.
Ant algorithms, which is a population-based approach, is applied to several NP-hard combinatorial optimization prob- lems. Ant algorithms, inspired from the behavior of real ant colonies [8], by their foraging behavior. Indirect communi- cation is one of the ideas of ant

algorithms and colony of agents are called artificial ants, based on pheromone trails. The pheromones are also used for communication by real ants. The pheromone trails are used to reflect their experience of distributed numeric information, modified by the ants while solving a particular problem. Ant Colony Optimization (ACO) meta-heuristic provides for most applications a framework to combinatorial optimization problems. The ant algorithms ap- plied to the TSP perfectly fit into the ACO meta-heuristic and, therefore, these algorithms also called as ACO algorithms [2]. In this paper, comparison between magnetic optimization algorithm and ant colony optimization is done for solving TSP. Here the comparison is based on the computation time for solving TSP rather than the optimal solution obtained using certain algorithm.

## II. Problem Definition

Traveling Salesman Problem is solved using two approaches: Magnetic Optimization Algorithm and Ant Colony Optimization Algorithm. Magnetic Optimization Algorithm, a novel heuristic optimization algorithm that has an inspiration of the principles of magnetic field theory. Ant algorithms have an inspiration of the behavior of real ants, by their foraging behavior. Based on pheromone trails, an indirect communication between colony of agents, called ants is the idea of ant algorithms.

### *A. Traveling Salesman Problem*

TSP is a problem of salesman and it is difficult to solve, but easy to describe. In graph theory, the problem is defined as for a given complete weighted graph, find the Hamiltonian cycle with the least cost. It is widely used in engineering applications where some industrial problems such as frequency assignment problems and machine scheduling can be formulated as a TSP. In [1] author proposed TSP as follows. TSP can be defined on a symmetric graph (or complete undirected graph), $G = (V, E)$ or asymmetric on G=(V,A). The vertex set is $V = \{1, .., n\}$, an edge set is $E = \{(i, j) : i, j \in V, \boldsymbol{i} < \boldsymbol{j}$ and an arc set is $A = \{(i, j) : i, j \in V, i \neq j\}$. A cost matrix $C = (cij)$ is defined on E or on A. The cost matrix satisfies the triangular inequality that is $\boldsymbol{cij} \leq cik + ckj$ for all i, j, k. In particular, in the planar region for which the vertices of points $Pi = (Xi, Yi)$ in the plane, and

$$cij = \sqrt{(Xi - Xj)^2 + (Yi - Yj)^2} \qquad (1)$$

is the Euclidean distance. The triangular inequality is also satisfied if $cij$ is the length of shortest path from $\boldsymbol{i}$ to $\boldsymbol{j}$ on $G$. The cost matrix $cij$ is calculated using Manhattan distance as

$$\boldsymbol{cij} = |Xi - \boldsymbol{Xj}| + |Yi - Yj| \qquad (2)$$

The mathematical formulation of a TSP for cost is said to be

$$c_{total} = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \times b_{ij} \qquad (3)$$

where

$$\sum_{i=1}^{n} b_{ij} = 1 \text{ for } i \in V, i \neq j \qquad (4)$$

$$\sum_{j=1}^{n} b_{ij} = 1 \text{ for } j \in V, j \neq i \qquad (5)$$

$$bij = 0 \text{ or } 1 \text{ for } (i, j) \in A \qquad (6)$$

## III. Related Works

One of the heuristic optimization algorithms is Magnetic Optimization Algorithm (MOA) which is inspired by magnetic field theory. It shows that MOA is useful for solving complex optimization problems.the binary version of MOA named BMOA is proposed in [6]. Developed two versions of BMOA with different topologies. To justify the performance of both versions, four benchmark functions employed, and its results compared with BPSO and GA. The results show that BMOA with fully-connected topology has advantage in binary search spaces, among heuristic optimization algorithms. In [1] authors have presented the application of MOA with voting modeling in TSP. The objective was to find the shortest distance for the salesman to visit all the cities once. Result obtained from the case study of [1] shows that this approach found a better solution than the solution suggested by TSPLIB.

Ant Colony optimization Algorithm which was introduced in early 1990s and is explained in [8]. Authors in [2] explains available ant colony optimization algorithms for the TSP. An outline of how ACO algorithms can be applied to that problem for the TSP. It briefly discusses local search for the TSP, experimental results are presented with MAX MIN Ant System. The proposed system in [4] is based on basic ACO algorithm along with well distribution strategy and information entropy which was conducted on the configuration strategy, updating the heuristic parameter in ACO to improve the performance in solving TSP. The main contribution was the study of the avoidance of stagnation behavior and premature convergence by using distribution strategy of initial ants and dynamic heuristic parameter updating based on entropy. Further a mer- gence of local search solution was provided. The experimental results and performance comparison showed in the proposed system of [4] reaches the better search performance over ACO algorithms do. In [7] explain the working of ACO algorithms which is easily understandable by solving TSP. Based on the experiments performed, it is concluded that the quality of solutions depends on the number of ants. The lower number of ants allows the individual to change the path much faster where as the higher number of ants in population causes the higher accumulation of pheromone, and thus an individual keeps the path with higher concentration of pheromone with a high probability.

## IV. Procedure

### *A. Magnetic Optimization Algorithm*

One of the fundamental forces of universe is Magnetic force. The force between electromagnetic particles that is inversely proportional to distance between them. Force between two par- ticle increases by decreasing the distance between them. The inspiration from electromagnetic force is the basic principle of Magnetic Optimization Algorithm.

A collection of search agents and magnetic particles is magnetic optimization algorithm. Magnetic particles have magnetic fields and mass proportional to fitness function. In each step, agents are attracted by electromagnetic forces. The forces' intensities are proportional to the distances and magnetic fields of agents.

Consider a system with $N$ agents. All agents are positioned randomly in a search space. The electromagnetic force on $(i,j)^{th}$ agent from $(u,v)^{th}$ agent at iteration $t$ is defined as:

$$f^k_{(i,j),(u,v)}(t) = \frac{B_{u,v}(t)}{D\left(x^k_{i,j}(t), x^k_{u,v}(t)\right)}\left(x^k_{u,v}(t) - x^k_{i,j}(t)\right) \qquad (7)$$

where at iteration $t$, $B_{u,v}(t)$ is the magnetic field of agent $u$, $v$, $x^k_{i,j}$ is the kth dimension of $(i,j)^{th}$ agent, $x^k_{i,j}$and $x^k_{u,v}$are kth dimensions of $(i,j)^{th}$ and (u, $v)^{th}$ agents, and D is the function calculating distance between agents.

The $D$ is calculated as:

$$D\left(x^k_{i,j}(t), x^k_{u,v}(t)\right) = \frac{1}{m}\sum_{k=1}^{m}\frac{x^k_{i,j}(t) - x^k_{u,v}(t)}{u_k - l_k} \qquad (8)$$

where $uk$ and $lk$ are the upper bound and lower bound of the $k^{th}$ dimension of the search space, respectively, and search space dimension is $m$.

Magnetic field of $(i,j)^{th}$ agent in iteration $t$ is calculated as:

$$Bi,j = Fitnessi,j\,(t) \qquad (9)$$

where $Fitness(t)$ is any fitness function to solve.

The resultant force on $(i,j)^{th}$ agent is calculated as given:

$$F^m_{i,j}(t) = \sum_{u.v \in N_{ij}} f^m_{(i,j),(u,v)}\,(t) \qquad (10)$$

where $Nij$ is the neighbor set of the agent $i,j$.
According to law of motion, acceleration of the agent is directly proportional to force and inversely proportional to its mass, so the acceleration of all agents are calculated as:

$$a^k_{i,j}(t) = \frac{F^k_{i,j}(t)}{M_{i,j}(t)} \tag{11}$$

where $Mi,j(t)$ is at iteration $t$, the mass of $(i,j)^{th}$ agent. Calculate the masses of agents as:

$$Mi,j(t) = \alpha + \rho \times Bi,j(t) \tag{12}$$

where $\alpha$ and $\rho$ are constant values.
The update velocity and position of the agents as follows:

$$vi,j(t+1) = rand \times vi,j(t) + ai,j(t) \tag{13}$$

$$xi,j(t+1) = xi,j(t) + vi,j(t+1) \tag{14}$$

where *rand* is random number in the interval [0,1].
MOA consists of 3 main parts: initialization, fitness evaluation, and improvement of agents [1]. Firstly initialize all agents with random values. Considered each agent as a candidate solution. Continuously run the steps until an end criterion is met. Define magnetic fields and masses for all agents. Then agents total forces and accelerations are calculated. And then the velocities and positions of each agents are updated.

## *B. Solving TSP using Magnetic Optimization Algorithm*

In this section, we show how TSP can be modeled using Magnetic Optimization Algorithm. For solving TSP using MOA, the algorithm is as given below:
*Algorithm 1 : Magnetic Optimization Algorithm*

```
1. Initialize TSP and MOA parameters, n, $c_{ij}$ , w(no. of iterations), h(no. of agents), $\alpha = 1$, $\rho = 1$.
2. Find distance matrix of cities.
3. Position the particles randomly
4. **while** not termination condition
5.          Find vote for each city by each particle randomly.
6.          Find solution suggested by each particle.
7.          Calculate solution for each solution.
8.                Store the distance as magnetic field, $b^z$.
9.          **if** the particle fitness is greater than global best **do**
10.               Store the solution and distance.
11.               Normalize magnetic field, $b^z$ as
                      $b^z_{ij} = \frac{b^z_{ij} - min_{i,j=1\to S}(B^t_{i,j})}{max_{i,j=1\to S}(B^t_{i,j}) - min_{i,j=1\to S}(B^t_{i,j})}$        (15)
            where $B_{i,j}$ is set of $b^t$ at $t^{th}$ iteration.
12.               Evaluate particle's mass, $m^z$ using (12).
13.               **for** all particles **do**
14.               Set magnetic force, $f^z = 0$.
15.               Find neighbors, $n^z$.
16.               **for** all neighbors **do**
17.                     Update magnetic force using (7).
18.               **end for**
19.               Update particle velocity, $v^z$ using (13).
20.               Update particle position, $x^z$ using (14).
21.               Perform correction if necessary.
22.         **end for**
23.   **end while**
24.  Display global best solution.
25.  Compute time.
```

*C. Ant Colony Optimization*

Single ant behavior is considered to solve complex problems. Instead, a colony of ants can solve complex problems thatcannot be handled by individuals. The ants have the ability to find shortest path from food source to the nest. That is, ants have ability to communicate indirectly using pheromone trails. This is the basic of ant colony optimization algorithm.

Initially, the artificial ant creates solution by traveling between vertices and by using probability specified by the amount of pheromone in each vertex, ant decides to choose next vertex to be visited. The ant remembers visited vertices to avoid sub-cycles. Ant returns by deterministic way and leaves pheromone trail on traveled edges. The left pheromone intensity decides quality of constructed solution. As time passes, the pheromone on each edge evaporates.

For solving of the problem of TSP, algorithm which used ant system (AS) which was presented in [2].

**Tour Construction:** Initially, ants are positioned randomly on each city. During each step, ant k performs a probabilistic action choice rule. The probability with which ant k, at city i currently, chooses next city j to go at the tth iteration is given by:

$$P_{i,j}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_k}[\tau_{il}(t)]^{\alpha}[\eta_{il}]^{\beta}} \qquad (16)$$

where $\eta_{ij}=1/d_{ij}$, $\beta$ *and* $\alpha$ are two parameters to determine, respectively, the heuristic information and the relative influence of the pheromone trail, and $N^k$ is the neighboring cities of ant $k$, that is, the set of cities which is not yet visited by ant *k*. The parameters $\alpha$ and $\beta$ has the following role. If $\alpha = 0$, selecting the closest city. If $\beta = 0$, only pheromone amplification is at work.

**Pheromone Update:** The pheromone trails are updated, after all ants have constructed tours. First, by a constant factor pheromone strength is lowered and then allow each ant add to pheromone on the visited arcs.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t) \qquad (17)$$

where $\rho$ is the pheromone trail evaporation that lies between 0 and 1. The parameter $\rho$, to avoid unlimited accumulation of the pheromone deposits and it enables to forget previously taken bad decisions. Pheromone strength decreases exponentially, if an arc is not chosen by the ants. $\Delta\tau^{k}(t)$ is the amount of pheromone ant $k$ puts on the visited arcs.

$$\Delta\tau^{k}(t) = \frac{1}{L^{k}(t)} \qquad (18)$$

if arc (i, j) is used by ant k, otherwise 0, where $L^k(t)$ is the length of the kth ant's tour. Arcs used by many ants and those contained in shorter tours will receive more pheromone and therefore are more likely to be chosen in future iterations. Since start of the algorithm, the best tour found is denoted by $T^{gb}$ (global-best tour). There are two types of pheromone update:

Global pheromone trail update:

After each iteration, pheromone is allowed to add by the global best ant. Update Pheromone as:

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + \rho.\Delta\tau_{ij}^{gb}(t) \qquad (19)$$

where

$$\Delta\tau_{ij}^{gb}(t) = \frac{1}{L^{gb}}$$

The pheromone update is applied to the arcs of the global-best tour, not to all the arcs. The parameter ρ is for pheromone evaporation.

- Local pheromone trail update:

  Additional to the global updating rule, the ants also does local update rule that is applied immediately after an arc is crossed during the tour construction:

$$\tau_{ij} = (1-\varepsilon).\tau_{ij} + \varepsilon.\tau_0 \qquad (20)$$

where $0 < \varepsilon < 1$ and $\tau_0$ are two ACO parameters. The local updating rule has an effect to make an already chosen arc less desirable for a following ants. The exploration of arc not yet visited is increased.

**Heuristic Parameter Updating (β):** When algorithm be- gins, information entropy is maximum. On each path as pheromone increases, entropy decreases gradually. Entropy will eventually reduce to 0, if it is not controlled and the pheromone on only one path will be maximum which will be as mistaken the final solution.
In the pheromone matrix each trail is a discrete random variable. The random variable entropy is defined as:

$$E(X) = -\sum_{i=1}^{r} P_i log P_i \qquad (21)$$

where Pi is the probability of occurrence of pheromone trails in the pheromone matrix. For symmetric n cities TSP, r = n(n−1)/2. The maximum entropy is as follows:

$$E_{max} = logr \qquad (22)$$

The heuristic parameter can be updated as the rule given by

$$\beta = \begin{cases} 5 & \text{threshold } X < E' \leq 1 \\ 4 & \text{threshold } Y < E' \leq \text{threshold } X \\ 3 & \text{threshold } Z < E' \leq \text{threshold } Y \\ 2 & 0 < E' \leq \text{threshold } Z \end{cases} \qquad (23)$$

$$E' = 1 - \frac{E_{max} - E_{current}}{E_{max}} \qquad (24)$$

E′ is calculated for current pheromone matrix entropy value and X, Y and Z are thresholds according to city size.

*D. Solving TSP using Ant Colony Optimization*

The steps taken for solving TSP using Ant Colony Optimization includes the following steps:

- Initialize: Allocate m ants randomly on n cities. The amount of pheromone on each edge is initialized to a tiny quantity.
- Pseudorandom proportional rule: Probability of an ant to move to city j from city i is based on priori available heuristic value and pheromone trail.
- Local Pheromone Update: After each construction step, all the ants perform updation.
- Computation of the optimal path
- Global Updating of pheromone: Update only in the optimal path.

Algorithm for solving TSP using ACO is described as follows:

Algorithm 2 : Ant Colony Optimization

```
1.  Initialize parameters for TSP and ACO.
2.  Initialize pheromone trails on each edge, $\tau ij$.
3.  Calculate maximum entropy, Emax = logr where r =n(n − 1)/2 and n is number of cities.
4.  while not termination condition
5.      Randomly position m ants on n cities.
6.      for each particle do
7.          Find unvisited cities.
8.              Ant moves in different route using probabilistic decision rule using (15).
9.              Ant completes one tour.
10.             Apply local pheromone update rule.
11.             Store the solution and distance.
12.     end for
13.     Apply global pheromone update rule.
14.     Calculate entropy value of current pheromone trails, $E_{current}$.
15.     Calculate $E'$ using (23).
16.     Heuristic parameter, $\beta$ is updated using (22).
17. end while
18. Display global best solution.
19. Compute time.
```

## V. Experimental Results

Input dataset for case study is taken from TSPLIB. It provides a number of datasets for solving TSP. Here, we have taken the sets namely, burma14, ulysses16 and bayg29.

TABLE I. BURMA14

| Iteration | Particle /Ant | Distance (MOA) | Distance (ACO) | Computation Time (MOA) (s) | Computation Time (ACO) (s) |
|---|---|---|---|---|---|
| 10 | 14 | 56.21 | 31.22 | 0.017318 | 0.088476 |
| | 10 | 49.42 | 31.22 | 0.013068 | 0.058507 |
| | 5 | 69.23 | 31.22 | 0.006479 | 0.030062 |
| 20 | 14 | 52.30 | 31.22 | 0.023520 | 0.144082 |
| | 10 | 49.42 | 31.22 | 0.022648 | 0.108659 |
| | 5 | 62.00 | 31.22 | 0.016466 | 0.059520 |
| 30 | 14 | 52.09 | 31.22 | 0.035630 | 0.225364 |
| | 10 | 49.42 | 31.22 | 0.026615 | 0.152643 |
| | 5 | 54.78 | 31.22 | 0.021157 | 0.084022 |

TABLE II. ULYSSES16

| Iteration | Particle /Ant | Distance (MOA) | Distance (ACO) | Computation Time (MOA) (s) | Computation Time (ACO) (s) |
|---|---|---|---|---|---|
| 10 | 16 | 99.18 | 55.61 | 0.01618 | 0.134162 |
| | 13 | 93.87 | 55.94 | 0.016066 | 0.078928 |
| | 11 | 89.65 | 55.94 | 0.015804 | 0.067428 |
| 20 | 16 | 95.53 | 55.32 | 0.033091 | 0.207363 |
| | 13 | 92.69 | 55.82 | 0.032398 | 0.178799 |
| | 11 | 89.65 | 55.21 | 0.016466 | 0.13024 |
| 30 | 16 | 85.30 | 55.32 | 0.046838 | 0.303615 |
| | 13 | 92.69 | 55.82 | 0.02627 | 0.239012 |
| | 11 | 89.65 | 55.21 | 0.044027 | 0.232875 |

TABLE III. BAYG29

| Iteration | Particle /Ant | Distance (MOA) | Distance (ACO) | Computation Time (MOA) (s) | Computation Time (ACO) (s) |
|---|---|---|---|---|---|
| 10 | 28 | 21416.12 | 10124.70 | 0.027322 | 0.467933 |
| | 27 | 21771.43 | 10332.06 | 0.026949 | 0.455241 |
| | 25 | 21395.15 | 10332.06 | 0.029791 | 0.428513 |
| 20 | 28 | 21416.12 | 10124.70 | 0.055019 | 0.953592 |
| | 27 | 19861.69 | 10332.06 | 0.052046 | 0.926285 |
| | 25 | 21067.86 | 10332.06 | 0.047581 | 0.13024 |
| 30 | 28 | 21416.12 | 10124.70 | 0.079973 | 1.326868 |
| | 27 | 19861.69 | 10332.06 | 0.071206 | 0.820827 |
| | 25 | 21067.86 | 10332.06 | 0.079659 | 1.275330 |

Fig 1 shows comparison of distanceand Fig 2 shows the comparison of computation time using MOA and ACO. Similarly Table 2 and Table 3 shows that of ulysses16 and bayg29 respectively. Fig 3 and Fig 4 shows comparison of distance and computation time respectively of ulysses16. Fig 5 and Fig 6 shows comparison of distance and computation time of bayg29 respectively. Hardware used to run the program has Processor type of Intel Core 2 Duo and speed 2000.0 MHz with Memory size 2048MB, memory type DDR2 SDRAM and memory speed 667 MHz. Table 1 shows the distance and computation time of burma14 using MOA and ACO.

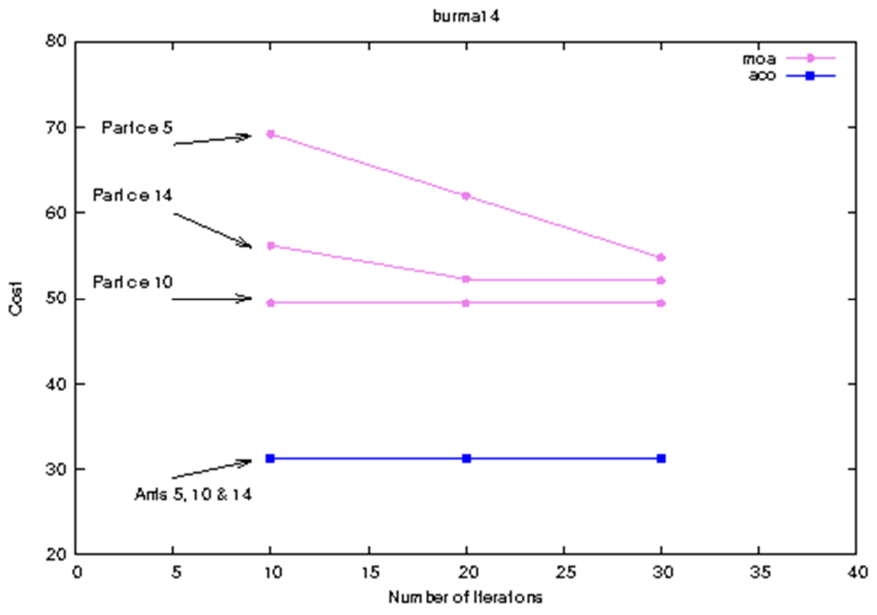

Fig. 1. Comparison of distance for ACO and MOA

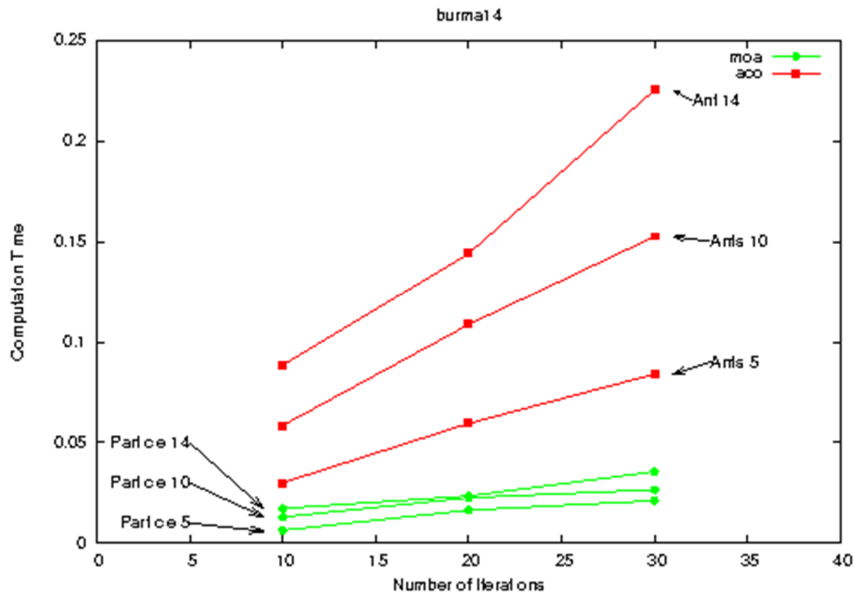

Fig. 2. Comparison of computation time for ACO and MOA

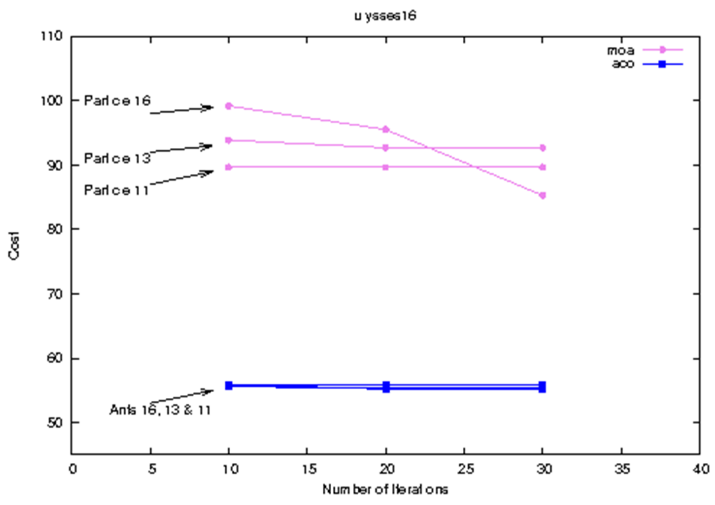

Fig. 3. Comparison of distance for ACO and MOA

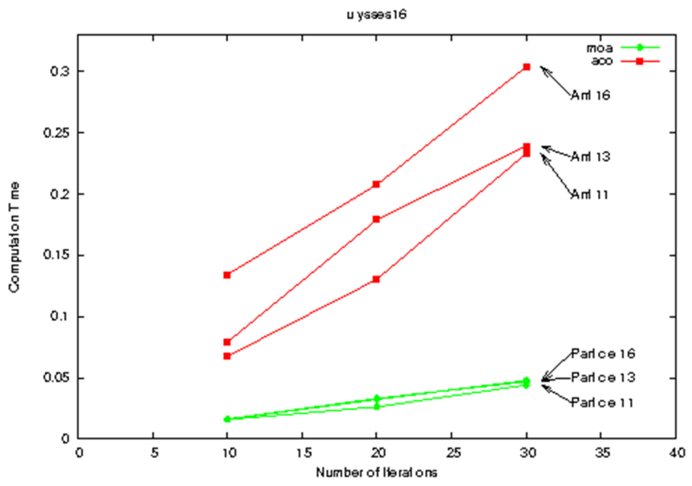

Fig. 4. Comparison of computation time for ACO and MOA

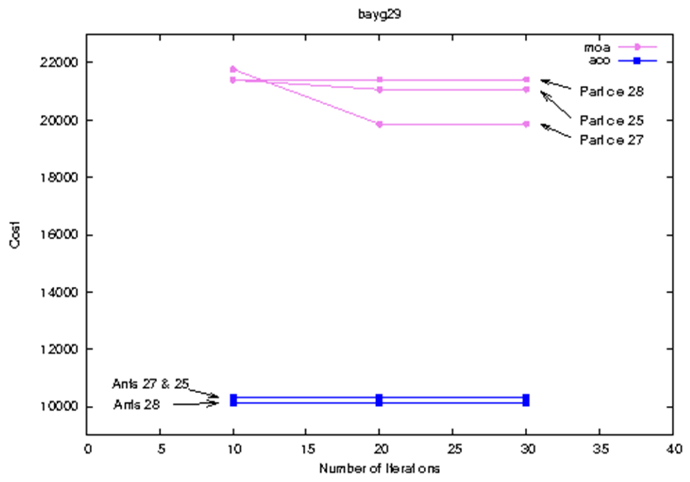

Fig. 5. Comparison of distance for ACO and MOA

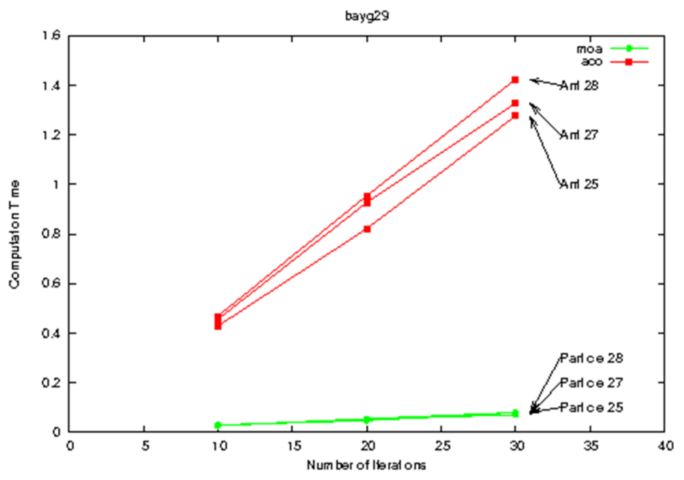

Fig. 6. Comparison of computation time for ACO and MOA

## VI. Inference

The following are the inferences obtained:

- Traveling Salesman Problem is solved using Magnetic Optimization Algorithm and Ant Colony Optimization Algorithm.
- Distance obtained for ACO is nearly optimal compared to MOA in all three cases of burma14, ulysses16 and bayg29.
- Time for computing the solution is much less for MOA.

## VII. Conclusion

The paper presents a comparison between ant colony opti- mization algorithm and magnetic optimization algorithm. ACO provides a better solution when compared to MOA. Instead, MOA obtains result in less computation time. For very large problems, the time for computing and obtain a near optimal result is the main factor. We conclude that MOA provides a near optimal solution in less computation time.

## References

[1] Mohd Muzafar Ismail, Muhammad Iqbal Zakaria, Amar Faiz Zainal Abidin, Juwita Mad Juliani, Asrani Lit, Seyedali Mirjalili, Nur Anis Nordin, Muhammad Faiz Mohamed Saaid. *Magnetic Optimization Al- gorithm Approach For Traveling Salesman Problem,* World Academy of Science, Engineering and Technology 62 2012

[2] Thomas STUTZLE and Marco DORIGO, *ACO Algorithms for the Trav- eling Salesman Problem,* IRIDIA, Universit Libre de Bruxelles, Belgium tstutzle,mdorigo@ulb.ac.be

[3] M. H. Tayarani-N and M.R. Akbarzadeh-T, *Magnetic Optimization Algo- rithms a new synthesis*, in IEEE Congress on Evolutionary Computation, 2008, pp. 2659 - 2664.

[4] Zar Chi Su Su Hlaing, May Aye Khine, *An Ant Colony Optimization Al- gorithm for Solving Traveling Salesman Problem,* University of Computer Studies, Yangon, IPCSIT vol.16 (2011), pp. 54-59.

[5] Sindhya K.Nambiar and Sumam Mary Idicula, *A Multi-Agent Routing System for Garbage Collection, Fifth International Conference on Advanced Computing (ICoAC), IEEE, 2013.*

[6] SeyedAli Mirjalili and Siti Zaiton Mohd Hashim, *BMOA: Binary Mag- netic Optimization Algorithm*, International Journal of Machine Learning and Computing, Vol. 2, No. 3, June 2012.

[7] Ivan Brezina Jr. Zuzana ikov, *Solving the Traveling Salesman Problem Using the Ant Colony Optimization*, Management Information Systems, Vol. 6 (2011), No. 4, pp. 010-014.

[8] M. Dorigo , V. Maniezzo and A. Colorni, *Ant System: An Autocatalytic Optimizing Process*, Technical Report 91-016

[9] Marco Dorigo, Luca Maria Gambardella, *Ant colonies for the traveling salesman problem,* TR/IRIDIA/1996-3 Universit Libre de Bruxelles, Belgium