ACO, Its Modification and Variants

Akash Tayal[1], Prerna Khurana[2], Priyanka Mittal[3], Sanjana Chopra[4]
[1](Electronics and Communication Engineering, Indira Gandhi Delhi Technical University for Women, India)
[2](Electronics and Communication Engineering, Indira Gandhi Delhi Technical University for Women, India)
3(Electronics and Communication Engineering, Indira Gandhi Delhi Technical University for Women, India)
4(Electronics and Communication Engineering, Indira Gandhi Delhi Technical University for Women, India)

ABSTRACT : *Ant colony optimization (ACO) is a P based metaheuristic algorithm which has been proven as a successful technique and applied to a number of combinatorial optimization problems and is also applied to the Traveling salesman problem (TSP). TSP is a well-known NP-complete combinatorial optimization (CO) problem and has an extensive application background.*
The presented paper proposes an improved version of Ant Colony Optimization (ACO) by modifying its parameters to yield an optimal result. Also this paper shows the experimental results and comparison between the original ACO and Modified ACO. Further this paper proposes two variants of ACO according to their specific application. Various city distributions have also been discussed and compared.

Keywords - *Ant Colony Optimization (ACO), Artificial Ants (AA), Combinatorial Optimization (CO), Particle Swarm Optimization (PSO), Travelling Salesman Problem (TSP)*

1. INTRODUCTION

There are numerous fields where optimization techniques can be useful for example science, engineering, economics and business. The real world problems are hard to solve, hence we make use of approximate algorithms with the aim of minimization of time, cost, and risk or the maximization of yield, quality, and efficiency.

Metaheuristics [2] solve instances of problems that are believed to be hard in general, by exploring the usually large solution search space of these instances. These algorithms achieve this by reducing the effective size of the space and by exploring that space efficiently. Metaheuristics serve three main purposes: solving problems faster, solving large problems, and obtaining robust algorithms. Moreover, they are simple to design and implement, and are very flexible. In the recent years, swarm intelligence, a new class of Metaheuristics, has emerged and attracted researchers' attention. Swarm intelligence imitates the social behavior of natural insects or animals to solve complex problems. Some commonly used swarm intelligence algorithms include ant colony optimization (ACO), particle swarm optimization (PSO) and artificial bee colony. ANT COLONY OPTIMIZATION is a P- based Metaheuristics technique which is a swarm intelligence technique, uses the concept of artificial ants acting as pheromone carrying agents, is commonly used in solving TRAVELLING SALESMAN PROBLEM. In this paper in section 2. ,we, first of all, have introduced the basic technique of ACO, describing various aspects of this algorithm following which we have defined our problem (TSP) in hand and the procedure to solve it in section 3. In section 4. We have shown how ACO is used for solving TSP. Section 5 mentions the selection of the optimal parameters of ACO and section 6. proposes variants of ACO .

Further section 7. gives the various city distributions and compares them. Section 8. concludes the entire paper indicating the pros and cons of the parameters obtained and the city distribution selected.

2. Theory and mathematical model of Ant Colony Optimization

2.1. Theory of Ant Colony Optimization

ACO is a population-based swarm intelligence algorithm and was proposed by Dorigo [3][4] and Gambardella. This algorithm has been inspired by the foraging behavior of real ant colonies. Ants coordinate their activities via stigmergy, a form of indirect communication mediated by modifications of the environment. When real ants collaborate to accomplish a task such as bringing food back to their nest they leave pheromones as they move back and forth between food sources and nest. Pheromone is a chemical that is olfactive and volatile in nature. This chemical substance has a decreasing action over time and the quantity left by one ant depends on the amount of food.

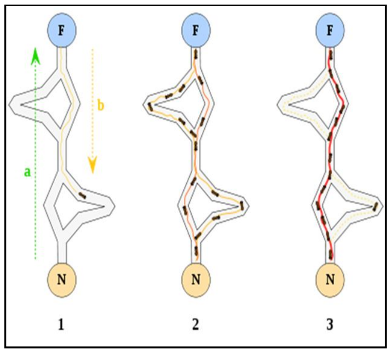

Figure 1: formation of shortest and optimal path by ants [10]

As shown in the Fig.1, there is an equal probability for the ants to choose either path a or b. As the central path is shorter and so requires less travel time, the ants will end up leaving a higher level of pheromones. The more the ants take this path, the higher the pheromone trail. Hence, there is an emergence of the shortest path as shown in Fig.1. This pheromone updating is an *autocatalytic Or a positive feedback* process which results in an optimal path.

2.2. Mathematical Model

The ACO algorithm uses the Artificial Ants (AA) in order to solve real world problems. The algorithm consists of basically two iterated steps: **solution construction** and **pheromone update** [6][7]

- **Solution construction**: Artificial ants create the solution by forming decision graphs in a probabilistic way. Decision graphs are constructed on the basis of pheromone trails which memorize the good generated solutions and these can be altered dynamically.
- **Pheromone trails:** Indeed, the pheromone trails memorize the characteristics of "good" generated solutions, which act as a positive feedback for the construction of new solutions by the ants. The pheromone trails change dynamically during the search to reflect the acquired knowledge. It represents the memory of the whole ant search process.
- **Pheromone update**: It is done in two phases:-
- An *evaporation phase* where the pheromone trails decreases automatically by a constant rate. Each pheromone value is reduced by a fixed proportion:

$$\boldsymbol{\tau_{ij}} = (\mathbf{1} - \mathbf{rr})\boldsymbol{\tau_{ij}},\ \forall\, i, j \in [1, n] \qquad \mathbf{(1)}$$

Where rr ∈ [0, 1] represents the reduction rate of the pheromone

A *reinforcement phase* where the pheromone trail is updated according to the generated solutions

```
Initialize the pheromone trails;
Repeat
    For each ant Do
        Solution construction using the pheromone trail;
    Update the pheromone trails:
        Evaporation;
        Reinforcement;
Until stopping criteria
Output: Best solution found or a set of solutions.
```

Figure 2: pseudo code of ACO

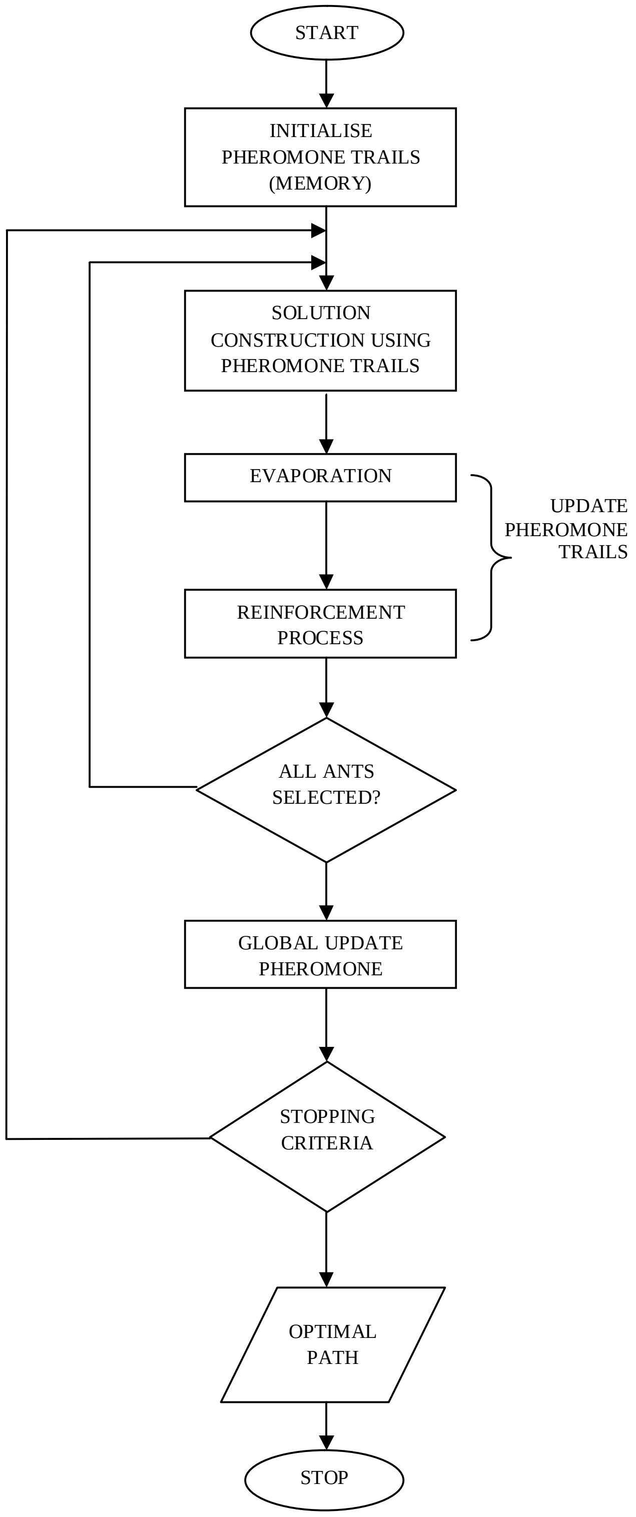

Figure 3: flowchart of ACO

3. Travelling Salesman Problem

3.1. Theory

Traveling salesman problem (TSP) [10] is a well known, popular and extensively studied problem in the field of combinatorial optimization. Its statement is apparently simple, but still it remains one of the most challenging problems in operational research. It is an optimization problem to find a shortest closed tour that visits all the given cities only once. It is known as a classical NP-complete problem [13], which has extremely large exploration spaces and is very complicated to solve. It is defined as *'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'* [8]

3.2. Mathematical Model

A complete weighted graph *G= (N, E)* can be used to represent a TSP, where N is the set of n cities and E is the set of edges (paths) fully connecting all cities. Each *edge (i,j)∈E* is assigned a cost *dij*, which is the distance between cities *I* and *j*. *dij* can be defined in the Euclidean space and is given as follows[14]:

$$\mathbf{d_{ij}} = \sqrt{(\mathbf{x_i} - \mathbf{x_j})^2 + (\mathbf{y_i} - \mathbf{y_j})^2} \qquad \mathbf{(2)}$$

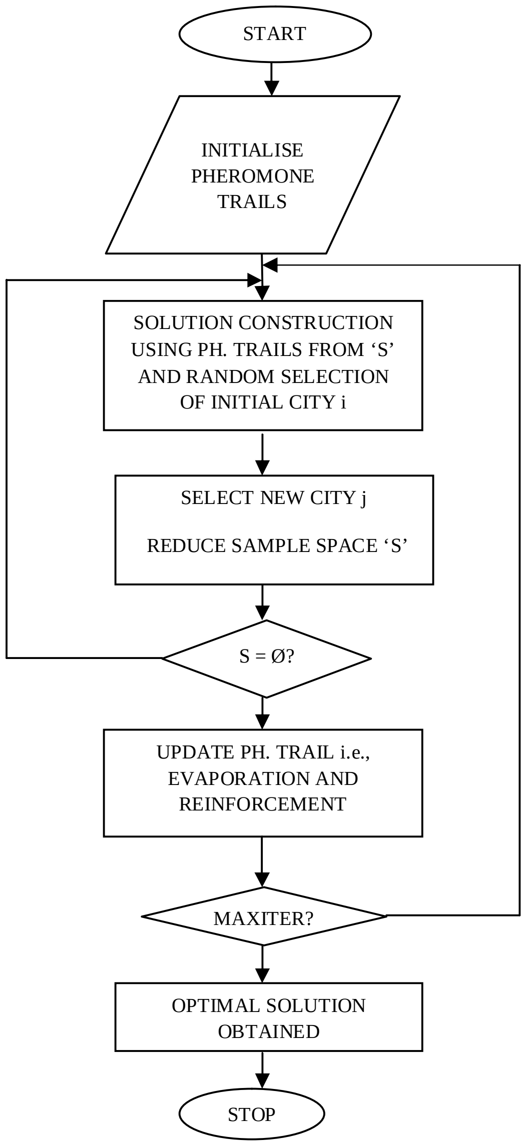
START
INITIALISE PHEROMONE TRAILS
SOLUTION CONSTRUCTION USING PH. TRAILS FROM 'S' AND RANDOM SELECTION OF INITIAL CITY i
SELECT NEW CITY j
REDUCE SAMPLE SPACE 'S'
S = Ø?
UPDATE PH. TRAIL i.e., EVAPORATION AND REINFORCEMENT
MAXITER?
OPTIMAL SOLUTION OBTAINED
STOP

Figure 4: flowchart of TSP

4. Procedure for solving TSP using ACO [9]

1) **Pheromone trails:** A pheromone τ_{ij} will be related with each edge (i, j) of the graph G. The pheromone information can be represented by an n × n matrix τ where each element τ_{ij} of the matrix expresses the attractiveness to have the edge (i, j) in the tour. The pheromone matrix is usually initialized by the same values. During the search, the pheromone will be updated to approximate the effectiveness of any edge of the graph.

2) **Solution construction:** Each ant will construct a tour in a stochastic way. Given an initial arbitrary city i, an ant will select the next city j with the probability

$$\mathbf{P_{ij}} = \frac{\tau_{ij}}{\sum_{k \epsilon S} \tau_{ij}} \qquad \forall\, \mathbf{j} \in \mathbf{S} \quad [9] \qquad \mathbf{(3)}$$

Where,

- The set **S** represents the not yet visited cities of the graph G.
- **τ** represent the pheromone strength

The ants may use a randomly selected initial city in the construction phase. The additional problem-dependent heuristic is defined by considering the values η_{ij} or the *visibility* equal to 1/dij where dij represents the distance between the cities i and j. The higher the heuristic value η_{ij} , the shorter the distance dij between cities i and j i.e. a more preferred node in terms of distance
Computing the decision transition probabilities P_{ij} is performed as follows:

$$\mathbf{P_{ij}} = \frac{\tau_{ij}^{\alpha} \times \eta_{ij}^{\beta}}{\sum_{k\,\epsilon\,S} \tau_{ik}^{\alpha} \times \eta_{ik}^{\beta}} \qquad \forall\, \mathbf{j} \in \mathbf{S} \quad [10] \qquad \mathbf{(4)}$$

Where,

- **η** represents the visibility of the next node or city
- **α** represents the *pheromone weighing* factor. If α = 0, the ACO algorithm will be similar to a stochastic greedy algorithm in which the closest cities are more likely selected.
- **β** represents *distance weighing factor*. If β = 0, only the pheromone trails will become dominant. In this case, a speedy emergence of stagnation may occur where all ants will construct the same suboptimal tour .

3) **Pheromone update :** Each ant will increment the pheromone related with the selected edges in a manner that is proportional to the quality of the obtained tour :[10]

$$\boldsymbol{\tau_{mn}} = (\mathbf{1} - \mathbf{rr})\boldsymbol{\tau_{mn}} + \sum_{k=1}^{Nants} \boldsymbol{\tau_{mn}^{k}} + \mathbf{e}\boldsymbol{\tau_{mn}^{elite}} \qquad \mathbf{(5)}$$

Where,

- τ_{mn} = pheromone laid by ant *k* between city *m* and city *n*
- **rr** = pheromone evaporation constant
- **e** = elite path weighting constant.

- τ_{mn}^{elite}= pheromone laid on the best path found by the algorithm to this point.

Then, good tours will emerge as the result of the collaboration between ants through the pheromone trails. The typical evaporation procedure is applied for the pheromone trails. For each edge, its pheromone τ_{ij} will evaporate as follows:[10]

$$\tau_{ij} = (1 - rr)\tau_{ij} \quad \forall \quad i, j \in [1, Ncity] \quad (6)$$

5. Optimal Parameters For ACO

Notations used

- Nants – number of ants
- Ncity – number of cities
- Phmone – initial pheromone level
- Iter – number of iterations
- rr – evaporation rate
- a – pheromone weighing factor
- b – distance weighing factor

5.1. Variation of Nants with Ncity

In this method we are observing the variation of Nants with Ncity in order to find the relationship between them.

Phmone=0.9, a=2, b=8, rr=0.25, iteration=1000

Table 1:Variation of Ncity with Nants

	Ncity = 30	**Ncity = 40**	**Ncity = 50**	**Ncity = 60**
Nants = 30	21.502	36.541	51.202	72.651
Nants = 40	29.928	49.389	69.162	101.385
Nants = 50	37.810	63.647	94.174	127.994
Nants = 60	44.517	76.705	112.591	155.476
Nants = 70	53.275	90.175	130.358	185.154
Nants = 80	61.874	106.692	150.885	211.827
Nants = 90	69.293	114.582	170.264	241.261
Nants = 100	77.771	131.713	190.051	270.209

The number of ants should be less than or equal to the number of cities for quick results however, the distance remains same. This maybe because as the number of ants are increased for a given number of cities only the computation time increases while the distance remains same.

Nants ≤ Ncity

5.2. Variation of Phmone with Nants and Ncity

Ncity= 30, a=2, b=8, rr=0.25, iter=1000

Table 2:Variation of Phmone with Nants

	Phmone = 0.1	**Phmone = 0.3**	**Phmone = 0.5**	**Phmone = 0.9**
Nants = 30	21.453	21.880	22.267	21.502
Nants = 40	29.755	29.108	29.228	29.928
Nants = 50	37.636	37.469	37.704	37.810
Nants = 60	44.911	44.680	45.064	44.517
Nants = 70	51.432	52.623	52.653	53.275
Nants = 80	60.760	61.011	61.692	61.874
Nants = 90	68.696	69.365	68.671	69.293
Nants = 100	78.070	77.542	77.525	77.771

Table 3: Variation of Phmone with Ncity

	Phmone = 0.1	**Phmone = 0.3**	**Phmone = 0.5**	**Phmone = 0.9**
Ncity = 30	21.257	22.472	21.121	21.502
Ncity = 40	37.747	37.437	36.744	36.541
Ncity = 50	50.288	53.245	50.229	51.202
Ncity = 60	71.963	72.675	71.651	72.650
Ncity = 70	93.455	95.231	95.860	95.828
Ncity = 80	125.606	123.994	125.168	124.732
Ncity = 90	151.891	151.566	148.995	149.485
Ncity = 100	197.594	196.071	196.029	199.461

The initial pheromone value of 0.3 displays better results in terms of time for most of the cases. As we are increasing the number of ants due to repeated deposition of pheromone the lesser initial value of pheromone displays better results.

Pheromone = 0.3

5.3.1. Time Variation of rr with Nants
Ncity=30, a=2 , b=8 , phmone=0.5 , iter=1000

Table 4:Time variation of rr with Nants

	rr = 0.25	**rr = 0.5**	**rr = 0.75**	**rr = 1**
Nants = 30	22.267	22.11	21.927	21.822
Nants = 40	29.228	29.626	29.350	28.308
Nants = 50	37.704	36.450	35.831	36.682
Nants = 60	45.064	44.760	44.685	43.465
Nants = 70	52.653	52.061	52.951	50.107
Nants = 80	61.692	60.952	60.660	60.385
Nants = 90	68.671	69.638	67.066	66.352
Nants = 100	77.525	75.993	74.433	75.493

When rr=1 , the initial pheromone level is totally not included i.e. no memory of initial pheromone levels**.** For a given number of ants as the rr is increased, the computation time is reduced.

5.3.2 Distance variation of rr with Nants
Ncity=30 , a=2 , b=8 , phmone=0.5 , iter=1000

Table 5:Distance variation of rr with Nants

	rr = 0.25	**rr = 0.5**	**rr = 0.75**	**rr = 1**
Nants = 30	44	44	45	45
Nants = 40	44	44	43	44
Nants = 50	44	43	44	45
Nants = 60	46	44	44	45
Nants = 70	43	43	43	44
Nants = 80	44	44	44	43
Nants = 90	44	44	46	43
Nants = 100	43	44	43	44

Distance increases with rr for a given number of Nants since the influence of initial pheromone levels decrease hence leading to randomness. To balance between both time and distance variation, optimum value of rr is taken to be 0.5.

rr = 0.5

5.4.1. Time variation of rr with Ncity
Nants=30 , a=2 , b=8 , phmone=0.5 , iter=1000

Table 6:Time variation of rr with Ncity

	rr = 0.25	**rr = 0.5**	**rr = 0.75**	**rr = 1**
Ncity = 30	22.121	21.848	22.135	21.822
Ncity = 40	36.744	36.373	37.645	34.495
Ncity = 50	50.829	52.761	50.847	47.308
Ncity = 60	71.651	69.491	73.071	67.025
Ncity = 70	95.860	95.131	99.055	89.910
Ncity = 80	125.168	116.754	123.522	110.979
Ncity = 90	148.995	140.528	141.054	130.294
Ncity = 100	197.929	191.937	197.285	154.284

5.4.2. Distance variation of rr with Ncity

Nants=30, a=2 , b=8 , phmone=0.5 , iter=1000

Table 7:Distance variation of rr with Ncity

	rr = 0.25	**rr = 0.5**	**rr = 0.75**	**rr = 1**
Ncity = 30	44	44	45	45
Ncity = 40	53	51	53	53
Ncity = 50	55	54	56	55
Ncity = 60	65	64	67	67
Ncity = 70	70	70	68	73
Ncity = 80	78	74	81	77
Ncity = 90	82	83	83	93
Ncity = 100	88	84	84	85

Optimum value of rr both in terms of time and distance can be seen as 0.5

rr = 0.5

5.5.1. Time variation of iter with Nants

Ncity=30 , a=2, b=8 , phmone=0.5, rr=0.25

Table 8: Time variation of iter with Nants

	iter = 10	**Iter = 100**	**Iter = 400**	**Iter = 800**	**Iter = 1000**
Nants = 30	0.2843	2.329	9.333	18.077	22.121
Nants = 40	0.364	3.082	11.698	23.294	29.228
Nants = 50	0.450	3.869	15.130	30.132	37.707
Nants = 60	0.520	4.496	17.741	35.460	45.064
Nants = 70	0.625	5.354	20.609	41.195	52.653
Nants = 80	0.707	6.521	25.207	49.768	61.692
Nants = 90	0.778	7.147	27.767	54.677	68.671
Nants = 100	0.881	8.071	31.577	62.843	77.525

5.5.2. Distance variation of iter with Nants

Ncity=30 , a=2, b=8 , phmone=0.5, rr=0.25

Table 9: Distance variation of iter with Nants

	iter = 10	**Iter = 100**	**Iter = 400**	**Iter = 800**	**Iter = 1000**
Nants = 30	47	47	46	46	44
Nants = 40	47	43	43	43	44
Nants = 50	45	43	44	44	44
Nants = 60	43	43	42	43	46
Nants = 70	44	43	43	43	43
Nants = 80	44	43	45	45	44
Nants = 90	45	45	43	44	44
Nants = 100	44	44	43	46	43

Time is increasing when the number of iterations is increased for a given number of ants. This is so because the number of loops has increased. Not much information can be drawn out from the distance variation graph since the number of cities is same; hence there is little variation with respect to ants.

5.6.1. Time variation of iter with Ncity

Nants=30, a=2, b=8, phmone= 0.5, rr=0.25

Table 10:Time variation of iter with Ncity

	iter = 10	Iter = 100	Iter = 400	Iter = 800	Iter = 1000
Ncity = 30	0.2843	2.329	9.333	18.077	22.121
Ncity = 40	0.449	3.969	15.061	28.129	36.744
Ncity = 50	0.645	5.133	19.851	38.703	50.829
Ncity = 60	0.887	7.715	28.253	54.911	95.860
Ncity = 70	1.151	9.950	37.091	76.570	97.929
Ncity = 80	1.515	12.595	47.471	87.054	125.168
Ncity = 90	1.752	14.456	53.44	106.609	148.995
Ncity = 100	2.303	19.508	67.52	126.234	197.929

5.6.2. Distance variation of iter with Ncity
Nants=30 , a=2 , b=8 , phmone= 0.5 , rr=0.25

Table 11:Distance variation of iter with Ncity

	Iter = 10	Iter = 100	Iter = 400	Iter = 800	Iter = 1000
Ncity = 30	47	47	46	46	44
Ncity = 40	55	51	54	53	53
Ncity = 50	56	56	55	55	55
Ncity = 60	67	67	67	66	88
Ncity = 70	74	71	74	68	70
Ncity = 80	81	79	77	75	78
Ncity = 90	87	83	87	89	88
Ncity = 100	94	87	86	88	88

In TABLE 10,we can see that the time is increasing as the numbers of iterations are increased since the number of loops is increasing. In TABLE 11, with increasing number of iterations the ACO yields more optimum results as for a given number of cities the distance is decreasing but after a certain number of iteration the result is getting static leading to just increased amount of time. It is observed that the max number of iterations should be of the order of 10 times the number of cities as it is giving favorable results.

5.7.1. Distance variation of a and b
Ncity=30,Nants=30,rr=0.25,phmone=0.5,iter=1000

Table 12:Distance variation of a and b

	a = 0.5	a = 1	a = 2	a = 3	a = 4	a = 5	a = 6
b = 0.5	87	70	48	52	59	65	79
b = 1	74	58	48	49	53	53	57
b = 2	59	48	46	49	48	51	50
b = 3	53	48	46	46	48	51	48
b = 4	50	50	45	45	51	47	52
b = 5	49	46	45	46	48	48	48
b = 6	46	46	45	46	45	46	45
b = 7	45	45	47	46	45	48	47
b = 8	47	45	44	46	47	46	45
b = 9	46	47	46	46	48	46	47
b = 10	46	46	46	47	46	46	44
b = 11	45	47	45	45	46	45	46
b = 12	45	45	44	45	46	46	44

5.7.2. Time variation of a and b
Ncity=30,Nants=30,rr=0.25,phmone=0.5,iter=1000

Table 13:Time variation of a and b

	a = 0.5	**a = 1**	**a = 2**	**a = 3**	**a = 4**	**a = 5**	**a = 6**
b = 0.5	28.126	25.343	23.604	25.775	24.367	24.297	22.907
b = 1	25.830	22.891	20.566	23,571	20.798	20.284	20.928
b = 2	26.563	20.733	18.907	22.774	21.299	20.819	20.963
b = 3	29.425	26.778	24.678	21.180	25.800	28.863	25.940
b = 4	28.7,83	26.042	23.710	28.774	27.607	26.594	26.234
b = 5	28.296	25.530	23.254	28.024	27.205	26.335	25.822
b = 6	28.300	24.967	22.873	27.375	26.899	26.111	26.028
b = 7	27.529	24.163	21.807	26.813	26.037	26.433	25.870
b = 8	27.248	23.775	21.901	26.584	26.586	27.487	27.036
b = 9	26.500	23.454	21.198	26.724	26.604	25.791	26.072
b = 10	26.451	23.319	21.061	26.724	26.394	25.937	26.001
b = 11	26.048	23.456	20.586	26.387	25.995	26.208	26.251
b = 12	25.844	22.910	20.782	26.233	25.839	26.394	26.344

It can be observed that the worst values are appearing for a, b<1 therefore we should keep the value of a and b >1. When a= 2 we are getting the minimum time for all the values of b. hence the best value for a is 2. As b is increasing graph converges quickly. Hence we conclude that value of b should be more. Also it has been observed that when value of b > a ACO yield in better results. In the time variation graph quick and optimal results are obtained when a =2 b= 2 When emphasis is on distance, a= 2 and b= 8 yields the favorable result with the value of dmin as 44.

Original results:

a=2, b=6, rr=0.5, phmone=0.1, Nants=Ncity=30, iter=600

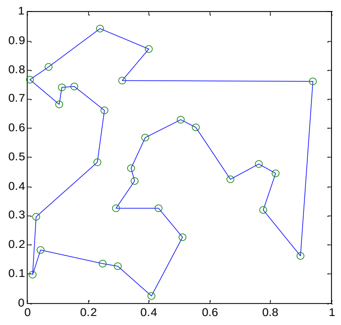

Figure 5: hamiltonian cycle

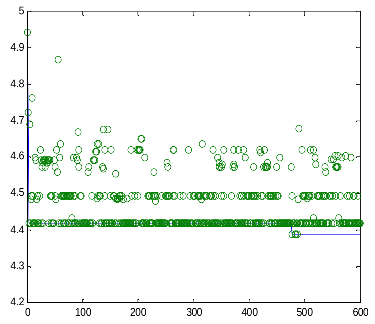

Figure 6: variation of dmin with iter

- Time Elapsed =14.32 s
- Dmin=44

Compared to original code, keeping the number of iterations same as 600

a=2, b=8, rr=0.5, phmone=0.5, Nants=Ncity=30

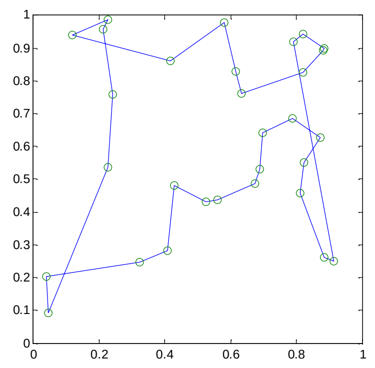

Figure 7: modified Hamiltonian cycle

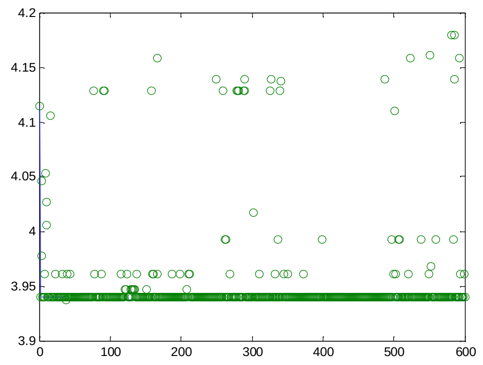

Figure 8: modified variation of dmin with iter

- Time Elapsed = 13.633s
- Dmin = 39

Thus compared to the original results we can see that there is a significant improvement in terms of distance from 44 to 39 giving a **11.36% decrease.** Also, in terms of time there is a decrease from 14.32s to 13.633s, which gives a **4.8% improvement.** Thus the change of parameters has resulted in achieving a relatively more optimized result in terms of both distance and time for cities=ants=30.

Graph showing the comparison between

1. a=0
2. b=0
3. original
4. modified

This graph clearly shows that the best results are obtained when the code is modified to give appropriate value of parameters and gives the worst result when b=0 i.e. the path formation depends entirely on pheromone levels.

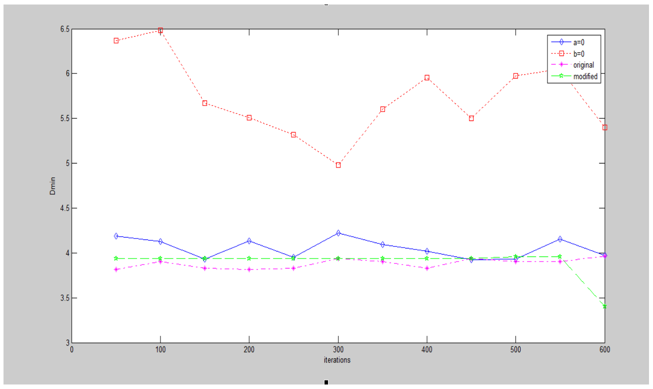

Figure 9: comparison between different parameter values

From the observations made above it can be concluded that the values of the parameters are:

a,b>1 also b>2a
here, a-2
b=8
0<rr<1 ; rr=0.5
0<phmone<1 ; phmone=0.5
Nants= Ncity
Iter> 10(Ncity)

Figure 10: optimal parameter values for ACO

6. VARIANTS OF ACO

6.1. NO HAMILTONIAN CYCLE (STRAIGHT LINE PATH)

When TSP is implemented using ACO we get a connected graph containing all the nodes to be visited. But the algorithm can also be modified in order to find the shortest distance between two nodes with the constraint that all other nodes have to be visited as well.

1) Mathematical Model

Algorithm for no Hamiltonian cycle (straight path)

```
Initialize the pheromone trails;
Repeat
        For each ant Do
                Initialization of tours such that no node is repeated;
                Solution construction using the pheromone trail;
                Update the pheromone trails:
                Evaporation;
                Reinforcement;
      Until Stopping criteria .
Output: Best solution found or a set of solutions
```

Figure 11: pseudo code for no Hamiltonian cycle ACO

2) Flowchart

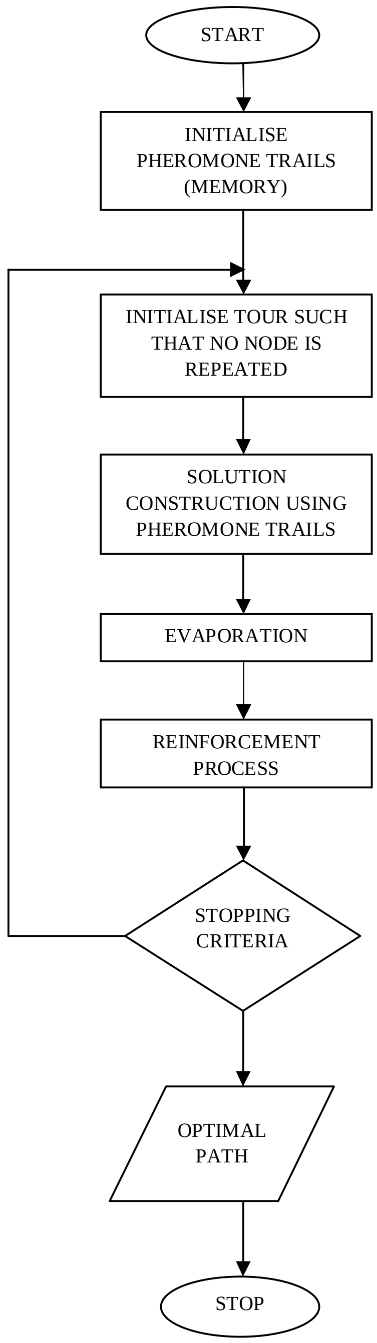
START
INITIALISE PHEROMONE TRAILS (MEMORY)
INITIALISE TOUR SUCH THAT NO NODE IS REPEATED
SOLUTION CONSTRUCTION USING PHEROMONE TRAILS
EVAPORATION
REINFORCEMENT PROCESS
STOPPING CRITERIA
OPTIMAL PATH
STOP

Figure 12: flowchart for no Hamiltonian cycle ACO

3) SIMULATION RESULT:

Ncity=30 , Nants= 30 , a=2 , b=8 , rr= 0.9, phmone=0.9 , Iter=1000

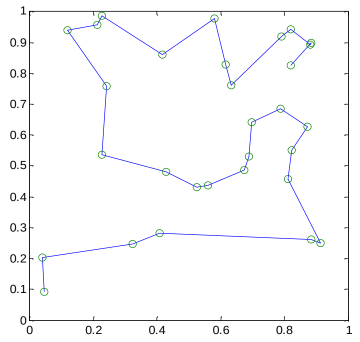

Figure 13: straight path

- Time Elapsed =24.117 seconds
- Dmin=39
- Converges in less than 50 iterations

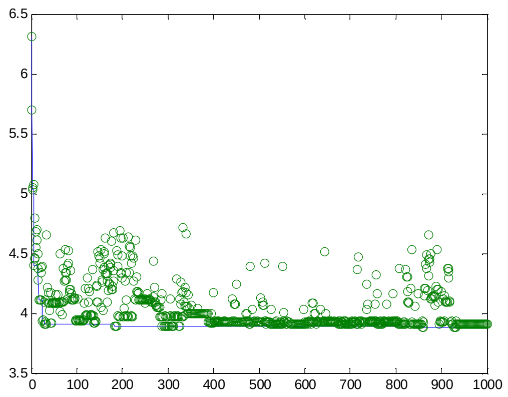

Figure 14 : convergence of distance with iteration

6.2. TIMING CONSTRAINTS

This is another variation of ACO where we have imposed timing restriction on our salesman wherein the salesman is expected to complete the tour in a given interval of time .If he is successful to do so , the output is displayed to the user otherwise no result is displayed.

1) ALGORITHM FOR TIME CONSTRAINT

```
Initialize the pheromone trails;
Repeat
        For each ant Do
          Solution construction using the pheromone trail;
                    Update the pheromone trails:
                              Evaporation;
                              Reinforcement;
                Until stopping criteria
        Check for satisfaction of timing condition.
            If time consideration is not met Output: no
                           solution.
   Else Output: Best solution found or a set of solutions.
```

Figure 15: pseudo code for time constraint ACO

2) FLOWCHART

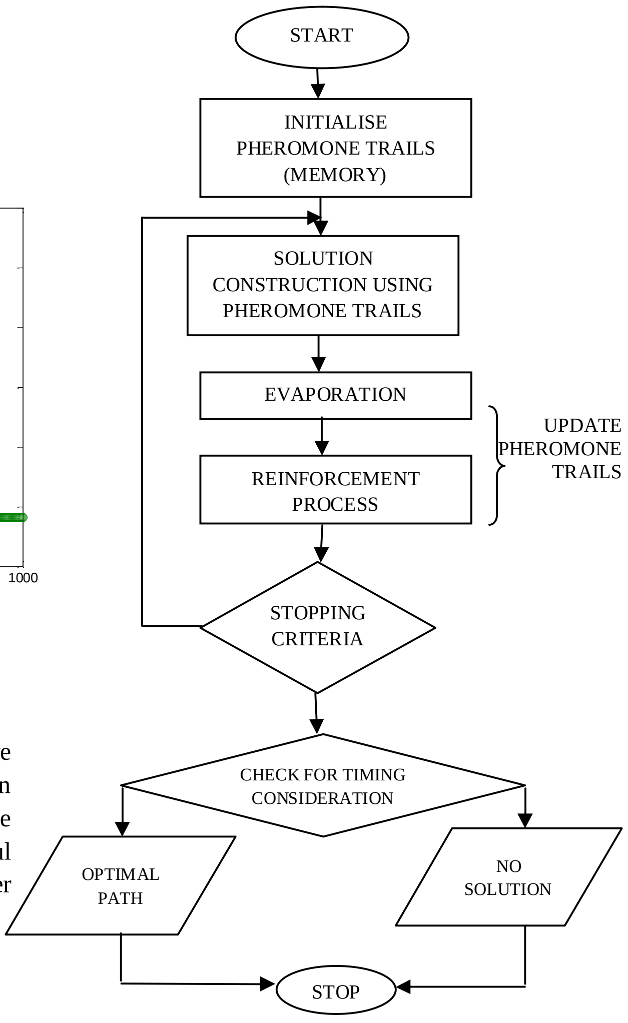
START
INITIALISE PHEROMONE TRAILS (MEMORY)
SOLUTION CONSTRUCTION USING PHEROMONE TRAILS
EVAPORATION
REINFORCEMENT PROCESS
UPDATE PHEROMONE TRAILS
STOPPING CRITERIA
CHECK FOR TIMING CONSIDERATION
OPTIMAL PATH
NO SOLUTION
STOP

Figure 16: flowchart of time constraint ACO

3) TABLE SHOWING TIME COSIDERATION RESULTS

Ncity=30 , Nants= 30 , rr=0.5, phmone =0.5 , a=2 , b=8 , iter =1000

Table 14:Time constraint ACO

TIME CONSTRAINT	Output
Time = 5	No result
Time = 10	No result
Time = 20	No result
Time = 30	22.2671
Time = 40	22.2671

4) SIMULATION RESULTS:

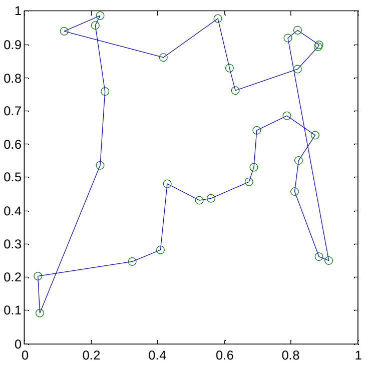

Figure 17: hamiltonian cycle

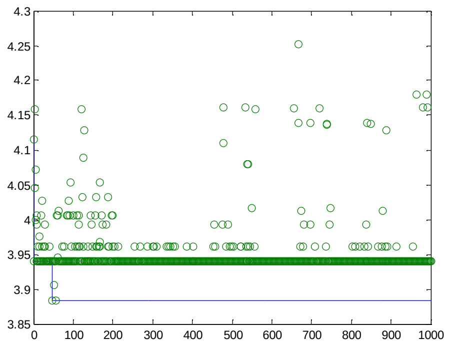

Figure 18: variation of dmin with iter

- Time Elapsed =22.2671seconds
- Dmin=44
- Converges in less than 100 iterations

7. VARIOUS CITY DISTRIBUTIONS

For all the previous cases we have considered a uniform distribution but we can also consider other distributions as well to distribute the city locations.

The various distributions considered are:

A. GAUSS NORMAL DISTRIBUTION

A normal Gaussian distribution pdf is

$$f(x, \mu, \sigma) = \frac{e^{\frac{-(x-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$$

Where μ is the mean or expectation of the distribution

And parameter σ is the standard deviation.

Here μ=0 and σ=1

B. EXPONENTIAL DISTRIBUTION

Pdf of exponential distribution is given as

$$f(x, \lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Where λ is the parameter of the distribution often called rate parameter. Here λ=1.

C. POISSON DISTRIBUTION

The pdf is given by

$$f(x, \lambda) = \frac{\lambda^x e^{-\lambda}}{x!}$$

Where λ is Poisson distribution parameter.

Here λ=5.

Tables showing comparison between the four distributions on the basis of variation of distances covered and time elapsed

Ncity=30 , Nants= 30 , rr=0.5, phmone =0.5 , a=2 , b=8,iter=600

Table 15:Comparision of various city distributions(distance)

Iter	Uniform distribution	Gauss normal distribution	Exponential distribution	Poisson distribution
100	2.1922	2.1023	2.1723	3.1561
200	4.4456	4.3918	4.3406	6.2608
300	6.4837	6.3519	6.3971	9.3476
400	8.6661	8.1074	8.3676	12.5140
500	10.9974	10.6782	10.5646	15.6277
600	13.2542	12.7943	12.7538	18.7444
700	13.5349	15.3065	14.8437	21.9669
800	17.8235	17.1270	16.9897	25.2293

Table 16:Comparison of various city distributions(time)

iter	Uniform distribution	Gauss normal distribution	Exponential distribution	Poisson Distribution
100	3.9374	16.3119	9.6465	35.6656
200	3.9374	16.9518	9.6465	35.6656
300	3.9374	16.5949	9.6433	35.6656
400	3.9374	16.1133	9.6433	35.6656
500	3.9374	15.0799	9.6433	35.6656
600	3.9374	13.3392	9.6433	35.6656
700	3.9374	13.9707	9.6433	35.6656
800	3.9374	17.9863	9.6433	35.6656

GAUSS NORMAL DISTRIBUTION

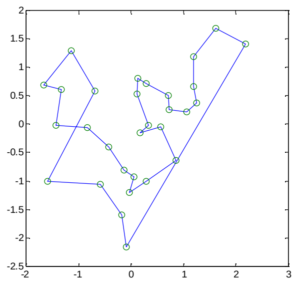

FIGURE 19: hamiltonian cycle

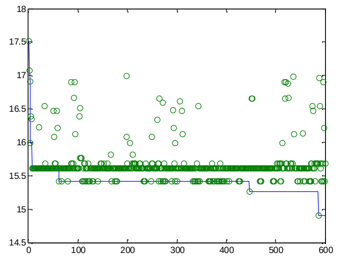

FIGURE 20:variation of Dmin with Iter

EXPONENTIAL DISTRIBUTION

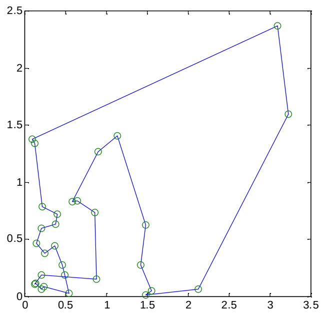

Figure 21: hamiltonian cycle

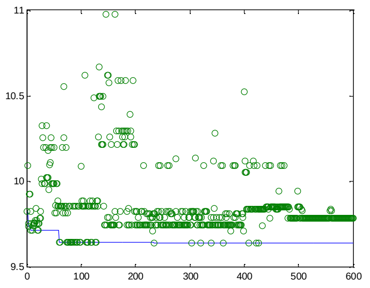

Figure 22: variation of Dmin with Iter

POISSON DISTRIBUTION

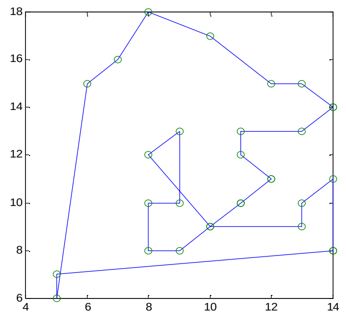

Figure 23: hamiltonian cycle

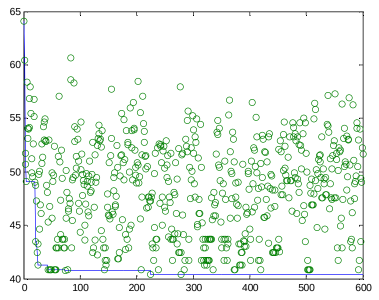

Figure 24: variation of Dmin with Iter

8. CONCLUSIONS

From the parameter variation section, it can be concluded that the values of the parameters are:

- a,b>1 also b>2a
- 0<rr<1 ;
- 0<phmone<1 ;
- Nants= Ncity
- Iter> 10(Ncity)

The following conclusions can be drawn from the distribution variation comparison table

- Time elapsed is highest for Poisson distribution as compared to other distributions.
- Time to converge is least for uniform distribution.
- Variation in distance is maximum for Gauss normal distribution.
- For exponential distribution the distance variation is less .However the time to converge is more as compared to uniform distribution.

REFRENCES

1. Magnus Erik Hvass Pedersen, Hvass Laboratories,
   "*Good Parameters for Particle Swarm Optimization"*, Technical Report no. HL10012010
2. M. Dorigo and T. Strutzle, "Ant colony optimization," *MIT Press*, Cambridge, MA, 2004
3. M. Dorigo, G.D. Caro, andL.M. Gambardella, "Ant algorithms for discrete optimization," *ArtificialLife*, vol. 5, no. 2, page 137, 1999
4. T. Stutzle and H.H. Hoos, "Max-min ant system," *Future Generation Comput. Syst*., vol. 16, no. 8,page 889, 2000
5. J. Kennedy and C. E. Russell, "Swarm intelligence," in Morgan Kaufmann, Academic Press, 2001
6. P. N. Suganthan, N. Hansen, J. J. Liang, et al., "Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization," Tech. Rep. 2005005, Nanyang Technological University, Singapore; IIT Kanpur, India, 2005.
7. M. I. Aouad, R. Schott, and O. Zendra, "A tabu search heuristic for scratch-pad memory management," in Proceedings of the International Conference on Software Engineering and Technology (ICSET '10), vol. 64, pp. 386–390, WASET, Rome, Italy, 2010.
8. S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, no. 4598, pp. 671–680, 1983. View at Scopus
9. "Practical GeniticAlgorithms" – Haupt and Haupt , second edition
10. "Metaheuristics – from Design to Implementation" by El-Ghazali Talbi
11. Vittorio Maniezzo, Luca Maria Gambarde, Fabio de Luigi.
    http://www.cs.unibo.it/bison/publications/ACO.pdf
12. Monash University CSE 460 lecture notes http://www.csse.monash.edu.au/~berndm/CSE460/Lectures/cse460-9.pdf
13. "Ant colonies for the traveling salesman problem" http://www.idsia.ch/~luca/acs-bio97.pdf
14. Dorigo, M. and Gambardella, L. M., Ant colonies for the travelling salesman problem, *Biosystems*, 43(2):73–81, 1997.
15. Chengming, Q., An ant colony algorithm with stochastic local search for the VRP, *8rd International Conference on Innovative Computing Information and Control*, Los Alamitos, CA, USA, pp. 464–468, IEEE Computer Society, 2008.
16. Lee, Z.-J., Su, S.-F., Chuang, C.-C., and Liu, K.-H., Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment, *Applied Soft Computing*, 8(1):55–78, 2008.
17. Jun-Qing Li, Q.-K. P. and Xie, S.-X., A hybrid variable neighborhood search algorithm for solving multi-objective flexible job shop problems, *Computer Science and Information Systems*, 7(4):907–930, 2010.
18. Negulescu, S., Dzitac, I., and Lascu, A., Synthetic genes for artificial ants. Diversity in ant colony optimization algorithms, *INT J COMPUT COMMUN*, ISSN 1841-9836, 5(2):216– 223, 2010.
19. Zhang, X., Duan, H., and Jin, J., DEACO: Hybrid ant colony optimization with differential evolution, *IEEE Congress on Evolutionary Computation*, 921–927, IEEE Computer Society, 2008.

20. Serbencu, A., Minzu, V., and Serbencu, A., An ant colony system based metaheuristic for solving single machine scheduling problem, *The Annals of Dunarea De Jos University of Galati*, 3:19–24, 2007.
21. Neumann, F., Sudholt, D., and Witt, C., Rigorous analyses for the combination of ant colony optimization and local search. *Ant Colony Optimization and Swarm Intelligence*, LNCS Berlin, Heidelberg, 5217:132–143, Springer-Verlag, 2008.
22. Gan, R., Guo, Q., Chang, H., and Yi, Y., Improved ant colony optimization algorithm for the traveling salesman problems, *Journal of Systems Engineering and Electronics*, 21(2):329– 333, 2010.
23. Jovanovic, R., Tuba, M., and Simian, D., Comparison of different topologies for islandbased multi-colony ant algorithms for the minimum weight vertex cover problem, *WSEAS Transactions on Computers*, 9(1):83–92, 2010.
24. Stutzle, T. and Dorigo, M., ACO algorithms for the traveling salesman problem, *Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications*, K Miettinen, P Niettaanmaki, M M Makela and J Periaux, editors, p. 500, Willey, 1999.
25. Stutzle, T. and Hoos, H. H., MAX-MIN ant system, *Future Generation Computer Systems*, 16(9):889–914, 2000.
26. Wong, K. Y. and See, P. C., A new minimum pheromone threshold strategy (MPTS) for max-min ant system, *Applied Soft Computing*, 9(3):882–888, 2009.
27. Huang, H., Yang, X., Hao, Z., and Cai, R., A novel ACO algorithm with adaptive parameter, *Computational Intelligence and Bioinformatics, LNCS*, 4115:12–21, Springer-Verlag Berlin Heidelberg, 2006.
28. White, C. and Yen, G., A hybrid evolutionary algorithm for traveling salesman problem, *IEEE Congress on Evolutionary Computation*, 2:1473–1478, IEEE Computer Society, 2004.
29. Duan, H. and Yu, X., Hybrid ant colony optimization using memetic algorithm for traveling salesman problem, *Approximate Dynamic Programming and Reinforcement Learning*, 92–95, IEEE Computer Society, 2007.
30. Reinelt, G., TSPLIB - a traveling salesman problem library, *ORSA Journal on Computing*, 3(4):376–384, 1991.
31. Jovanovic, R. and Tuba, M., Ant colony optimization algorithm with pheromone correction strategy for the minimum connected dominating set problem, *Computer Science and Information Systems (ComSIS)*, 10(1):133–149, 2013, DOI:10.2298/CSIS110927038J.
32. Jovanovic, R., Tuba, M., and Simian, D., An object-oriented framework with corresponding graphical user interface for developing ant colony optimization based algorithms, *WSEAS Transactions on Computers*, 7(12):1948–1957, 2008.
33. E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy-Kan and D.B. Shmoys, "The Travelling Salesman Problem," *New York: Wiley*, 1985.
34. J.L. Bentley, "Fast algorithms for geometric traveling salesman problems," *ORSA Journal on Computing*, Vol. 4, pp. 387–411, 1992.
35. M. Dorigo and L.M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Transactions on Evolutionary Computation*, Vol.1, No.1, pp. 53-66, 1997.
36. M. Dorigo, V. Maniezzo and A.Colorni, "The ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, Vol. 26, No. 2, pp. 1-13, 1996.
37. Matthijs den Besten, Thomas Stützle, and Marco Dorigo. Ant colony optimization for the total weighted tardiness problem. In PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, pages 611–620, London, UK, 2000.Springer-Verlag. ISBN 3-540-41056-2.
38. Thomas Eiter and Georg Gottlob. Hypergraph transversal computation and related problems in logic and ai. In JELIA *'02*: Proceedings of the European Conference on Logics in Artificial Intelligence, pages549–564, London, UK, 2002. Springer-Verlag. ISBN 3-540-44190-5.
39. Martin Charles Golumbic. Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol *57*). North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands,2004. ISBN 0444515305.
40. Christine Solnon. Solving permutation constraint satisfaction problems with artificial ants. In in Proceedings of ECAI'*2000*, IOS,pages 118–122. Press, 2000.
41. G. Bilchev and I. Parmee (1995) The Ant Colony Metaphor for Searching Continuous Design Spaces, *Proceedings of the AISB Workshop on Evolutionary Optimization*, Berlin, pp. 25-39.
42. L. Kuhn (2002) Ant Colony Optimization for Continuous Spaces, thesis, Department of Information Technology and Electrical Engineering, University of Queensland, Australia.
43. M. Matsumoto and T. Nishimura (1998) Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, *ACM Transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3-30.
44. E. Chong and S. Zak (2001) *An Introduction to Optimization, 2nd Edition*, New York, Wiley-Interscience.
45. W. Sun and Y. Yuan (2006) *Optimization Theory and Methods: Nonlinear Programming*, New York, Springer.
46. E. Bonabeau, M. Dorigo, and G. Theraulaz (1999) *Swarm Intelligence: From Natural to Artificial Systems*, New York, Oxford University Press
47. R. Marler and J. Arora (2004) Survey of Multi-Objective Optimization Methods for Engineering, *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369-395.
48. K. Miettinen (1998) *Nonlinear Multiobjective Optimization*, New York, Springer.
49. Y. Donoso and R. Fabregat (2007) *Multi-Objective Optimization in Computer Networks Using Metaheuristics*, Chicago, Auerbach.
50. I. Das and J. Dennis (1997) A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problems, *Structural and Multidisciplinary Optimization*, vol. 14, no. 1, pp. 63-69.
51. J. Sanchis, M. Martinez, X. Blasco, and J. Salcedo (2007) A New Perspective on Multiobjective Optimization by Enhanced Normalized Normal Constraint Method, *Structural and Multidisciplinary Optimization*, Springer Berlin.

52. M. Matsumoto and T. Nishimura (2000) Dynamic Creation of Pseudorandom Number Generators, *Monte Carlo and Quasi-Monte Carlo Methods 1998*, Springer, pp. 56-69.
53. I. Egorov (2003) IOSO NM Version 1, User Guide, IOSO Technology Center.
54. M. Kong and P. Tian (2005) A Binary Ant Colony Optimization for the Unconstrained Function Optimization Problem, *Lecture Notes in Computer Science*, Vol. 3801, pp. 682-687.
55. Q. Zhang, "Research on Ant Colony Algorithm and its Applications", Computer Knowledge and Technology, Vol. 5, No.9, 2009, pp. 2396-2398.
56. C. Blum, A. Roli and M. Dorigo, "HC-ACO: The hyper-cube framework for Ant Colony Optimization", *Proceedings of MIC 2001-meta-heuristics International Conference*, Porto, Portugal, July 16-21, 2001, pp.399-404.
57. L. Chen and Z. Pan, "Ant colony optimization approach for test scheduling of system on chip", Journal of Chongqing University of Posts and Telecommunications, Vol.21, No.2, 2009, pp.212-217.
58. M. L. Spangler, K. R. Robbins, J. K. Bertrand and M. Macneil, "Ant colony optimization as a method for strategic genotype sampling", Animal genetics, Vol. 40, No. 3, 2009, pp.308-314.
59. W. Tsai and F. Tsai, "A New Approach for Solving Large Traveling Salesman Problem Using Evolutionary Ant Rules," *IJCNN 2002, IEEE.*
60. H. Md. Rais, Z. A. Othman, and A. R. Hamdan, "Improved dynamic ant colony system (DACS) on symmetric Traveling Salesman Problem (TSP) ," *International Conference on Intelligence and Advanced Systems, IEEE*, 2007.
61. J. Han and Y. Tian, "An improved ant colony optimization algorithm based on dynamic control of solution construction and mergence of local search solutions," *Fourth International Conference on Natural Computation*, IEEE, 2008.
62. M. Colpan, "Solving geometric tsp with ants," the pennsylvania state university, 2005
63. C.-M. Pintea and D. Dumitrescu, "Improving ant system using a local updating rule," *Proceedings of the Seventh International Symposium and Numeric Algorithms for Scientific Computing (SYNASC'05), IEEE* 2005.
64. R. Gan, Q. Guo, H. Chang, and Y. Yi, "Improved ant colony optimization algorithm for the traveling salesman problems," *Jouranl of Systems Engineering and Electronics*, April 2010, pp 329-333.
65. C.-X. Wang, D.-Wu. Cui, Y.-K. Zhang, and Z.-R. Wang, "A novel ant colony system based on delauney triangulation and self-adaptive mutation for tsp," *International Joural of Information Technology*, Vol.12, No.3, 2006.
66. Z. A. Othman, H. Md. Rais, and A. R. Hamdan," Strategies DACS3increasing its performances," *European Journal of Scientific Research*, 2009.
67. K. S. Hung, S. F. Su, and S. J. Lee, "Improving ant colony optimization for solving traveling salesman problem," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 2007.
68. D. X. Yu, "Hybrid ant colony optimization using memetic algorithm for traveling salesman problem," in *Proceedings of the 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning* (ADPRD 2007).
69. L. Min and J. Yant, "A shortest path routing based on ant algorithm,"\ *Journal of Communication and Computer*, ISSN1548-7709, USA, September 2005.
70. Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The ant system: Optimization by a colony of coorperating agents. IEEE Trans. on Systems, Man and Cybernetics- Part B, 26(1):29–41, 1996.