Network and Complex Systems
ISSN 2224-610X (Paper) ISSN 2225-0603 (Online)
Vol.3, No.10, 2013

Computational Numerical Solution for Traveling Salesman Problem

D. G. Iyanda, (Corresponding Author)

Department of International Co-operation, State Planning Commission of Osun,
New Secretariat, P. M. B. 4435, Osogbo, Nigeria.
Tel. Phone: 0806-634-6384 E-mail: dgiyanda@fastmail.fm

L.A. Ogundele,
Department of Computer Science, Osun State College of Education, Ilesa, Osun State, Nigeria.
Tel. Phone: 0806-231-2269 E-Mail: lukman_ogundele@yahoo.com

Ojo, O.S.
Department of Mathematics, Osun State College of Education, Ilesa, Osun State, Nigeria.
Tel. Phone: 0803-727-4282 E-Mail: ojolayi9@gmail.com

Olatunji, O,
Department of Mathematics, Osun State College of Education, Ilesa, Osun State, Nigeria.
Tel. Phone: 0803-377-7923 E-Mail: olusuji4me@yahoo.com

Abstract

This paper examined and analysed the desire of Traveling Salesman Problem (TSP) to find the cheapest way of visiting all given set of cities and returning to the starting point. We presented a unique decomposition approach model for TSP in which the requirements and features of practical application in communication network, road transportation and supply chains are put into consideration. We used a Mathematical Modeling solution with the application of Ant Colony Search Algorithm (ACSA) approach for result computation.

In our approach, different Agents were created for difference purposes. *Information agent* gathered information about best tour and detected the *solution agent* that arrived at a given point with information message containing details of where the *solution agent* has come from as well as best tour cost. The *place ant* performs local pheromone decay on the relevant links. This help to avoid random visit to irrelevant edges and allows the *place ant* to calculate the cost of tour of all *place ants* including the latest pheromone level on the links to each of the *place ants.* The *solution agent* uses available information to decide which node to visit next and informs the *place ant* of its decision to move to a given destination and update better tour previously sampled while information about where to go next also obtained. The *place ant* updates its pheromone value for that link using the equivalent of the algorithm for local pheromone update. The cycle continues until *solution agent* arrives at its destination.

The main advantage of our approach is that it permits the use of mixed integer programming and combinatorial optimization techniques to compute real optimal routing path, solving the problem in practice by returning actual shortest route with its numerical value and not the best effort result as provided by some previous models and analytical methods.

The implementation was carried out using C# programming language. Data used were generated and the performance evaluation of the model was carried out through simulation using Matlab 7.0. The result shows that by considering all possible paths between a node as *the source* and another as *the destination*, all possible routes for a particular journey with shortest route in each case were generated.
Keywords: *Ant Colony, Combinatorial Optimization, Mixed Integer Programming, Pheromone, Search Algorithm and Traveling Salesman.*

Introduction

The Travelling Salesman Problem (TSP) is a problem in combinatorial optimization studied in operations research and theoretical computer science. Given a list of cities and their pairwise distances, the task is to find a shortest possible tour that visits each city exactly once. The problem was first formulated as a mathematical problem in 1930 and is one of the most intensively studied problems in optimization. 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. The TSP has several applications even in its purest formulation, such as planning, logistics, and the manufacture of microchips. Slightly modified, it appears as a sub-problem in many areas, such as DNA sequencing. In these applications, the *city* represents, for example, customers, soldering points, or DNA fragments, and the *distance* represents travelling times or cost, or a similarity measure between DNA fragments.

In many applications, additional constraints such as limited resources or time windows make the problem considerably harder. In the theory of computational complexity, the decision version of TSP belongs to the class of complete Network Problems (NP). Thus, it is assumed that there is no efficient algorithm for solving TSPs. In other words, it is likely that the worst case running time for any algorithm for TSP increases exponentially with the number of cities, even some instances with only hundreds of cities will take many CPU years to solve. As explained by Lenin and Mohan (2006), ant colony search algorithms, to some extent, mimic the behaviour of real ants. Real ants are capable of finding the shortest path from food sources to the nest without using visual cues. They are also capable of adapting to changes in the environment; for example, finding a new shortest path once the old one is no longer feasible due to a new obstacle.

Real ants use pheromone to communicate. They initially wander randomly searching for food, when they eventually find it they return to the nest laying the pheromone on the ground. Other ants that smell pheromone are more likely to follow the trail and reinforce the pheromone while returning with the food to the nest. The other important property of the pheromone is its evaporation, long trails not used by many ants start to disappear. This always leads to using only the shortest paths to food source. Brueckner (2002) first introduced the concept of a pheromone infrastructure in the context of manufacturing control. Parunak et al. (2004) observed that place agents manage four pheromone functions, namely: aggregation, evaporation, propagation and sensing. According to Ridge et al. (2005), pheromone infrastructure can be used to methodically move the Ant Colony Optimization algorithm into a probability and deterministic environment. Since the aim of the Traveling Salesman Problem (TSP) is to find the cheapest way of visiting a

given set of cities where the cost of travel between each pair of them is given, including the return to the starting point, TSP is a very good example of a larger class of problems known as combinatorial optimization problems. Thus TSP belongs to the so called NP (nondeterministic polynomial) hard complexity class. TSP is a well studied example of NP-hard problem, Applegate et al. (2006).

An algorithm can be considered as an effective one if it has a polynomial function of the problem size *n*, that is, for large values of *n*, the algorithm runs in time at most *Knc* for some constant number *K* and *c*. If an efficient algorithm i.e., an algorithm that will guarantee to find the optimal solution in polynomial number of steps can be found for the traveling salesman problem, then efficient algorithms could be established for all other problems in the NP class. Since the original formulation of the problem is aim to find the "cheapest" tour, thus the cost matrix that represents the distances between each pair must be determined by calculating the actual costs of transportation processes. The costs of transportation consist of two main elements: costs proportional to transit distances (km) and costs proportional to transit times. Obviously the physical distances can be considered as constant values in a given relation. Furthermore the actual costs are rarely constant and predictable, so fuzzy cost coefficient can be applied in order to represent the uncertainty, Ammar and Youness (2005), Kikuchi and Chakroborty (2006).

On the other hand, in real road networks the actual distance between two points often alters from the euclidean distance. Considering these characteristics the original TSP should be reconstructed, so that realistic solutions can be developed. For solving the road-transport TSP (RTTSP), we apply Ant Colony Search Algorithm (ACSA) since that algorithm is suitable for global optimization of even non-linear, high dimensional, multi-modal, and discontinuous problems. As numerical example, a modified road-transport TSP (RTTSP) instance is considered, in which the elements of cost matrix are dependent on the steps they are selected to carry on with. Rather than enumerating all possibilities, successful algorithms for solving the TSP problem shall capable of eliminating most of the roundtrips without ever actually considering them. As indicated by Dorigo et al (1996), these applications include but not limited to the following: Minimum Spanning Tree Problem (MST), Resource Location and Discovery in peer-to-peer network system, Vehicle Routing, Sequential Ordering, Connection-oriented Network Routing, Connectionless Network Routing, Optical Network Routing, Constraint Satisfaction, 2D-HP Protein folding, etc.

Previous related models

Many heuristic searches and algorithms have been suggested with their practical importance and wide range of application in Applegate et al. (2006), Ding et al (2007):, Shi et al.(2007), Bontoux and Feillet (2008), Yu-Wan et al (2007) and Dorigo (1992). However, the question whether or not there is a good algorithm for the TSP has not been settled. Nowadays numerous successful implementations of the Ant Colony Optimization (ACO) metaheuristic are available and properties of ants behaviour are being used to resolve many Optimization Problems that can be reduced to finding shortest paths. Dorigo, et al. (1996) observed that ACO had been applied to many different combinatorial optimization problems but careful study of these models revealed that they are just *best effort* results.

Ant colony scheme

The Ant Colony Optimization (ACO) concept is applied in this work to solve problem of TSP over a road network environment by creating a population of artificial ants that searches for optimal solutions (shortest paths) according to the problem's constraints. Artificial ants are used as agents that imitate the behaviour of real ants. However, the artificial Ant Colony System

(ACS) used in this work has the following differences in comparison with a real Ant Colony System (ACS), (i) artificial ants have memory; (ii) they are not completely blind; and (iii) they live in an environment where time is discrete. In addition, the artificial ACS has the following characteristics adopted from real ACS, (i) artificial ants have a probabilistic preference for routes with a larger amount of pheromone (ii) shorter routes tend to have larger rates of growth in their amount of pheromone and (iii) the ants use an indirect communication system based on the amount of pheromone deposited in each route.

The key idea is that, when a given ant has to choose between two or more routes, the route that was more frequently chosen by other ants in the past will have a greater probability of being chosen by the ant. Therefore, trails with greater amount of pheromone are synonyms of shorter routes. In essence, an ACS iteratively performs a loop containing two basic procedures, namely: (i) a procedure specifying how the ants construct or modify solutions of the problem being solved and (ii) a procedure to update the pheromone trails. The construction/modification of a solution is performed in a probabilistic way. The probability of adding a new item to the current partial solution is given by a function that depends on a problem-dependent heuristic and on the amount of pheromone deposited by ants on this trail in the past. The updates in the pheromone trail are implemented as a function that depends on the rate of pheromone evaporation and on the quality of the produced solution. To realize this, the following are well defined (i) appropriate representation of the problem which allows the ants to incrementally construct/modify solutions through the use of a probabilistic transition rule based on the amount of pheromone in the trail and on a local heuristic (ii) heuristic function that measures the quality of items that can be added to the current partial solution (iii) method to enforce the construction of valid solutions, i.e. solutions that are legal in the real-world situation corresponding to the problem definition.(iv) rule for pheromone updating which specifies how to modify the pheromone trail (v) probabilistic rule of transition based on the value of the heuristic function and on the contents of the pheromone trail.

As noted by Dorigo et al. (1996)], ants while almost blind, manage to establish shortest route paths from their colony to feeding sources and back only by communicating information by laying on the ground a substance called pheromone. This behaviour has inspired what is known as Ant Colony System (ACS) algorithms that use colonies of artificial ants.

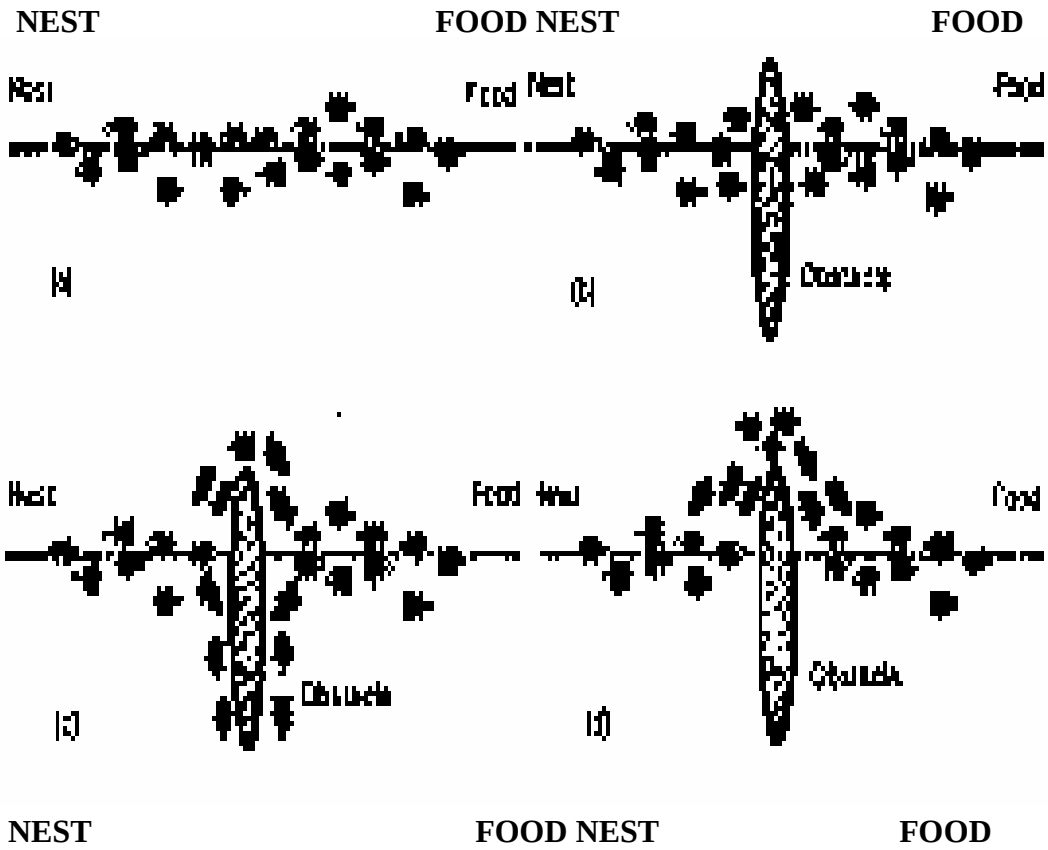

Figure 1: Artificial ant colony
Source: Lenin, K. and M. R. Mohan,(2006)

In our model, ants are send in regular intervals to randomly select path similar to the routing algorithm used in Dorigo et al. (1996) framework. The different between our model and Dorigo et al. (1996) is the intelligent ability of ants through which they are able to collect delay as well as congestion information and use "pheromone data" left by previous ants to smooth their movement. After reaching target destination they return on the same path and update visited links with information they collected on the way. The positive feedback is the trail of pheromone left by ants that had found the food. So each pheromone trail leads to some destination. The negative feedback is the evaporation of the pheromone with each iteration of the algorithm. Smaller amount of pheromone attracts ants with less probability, therefore longer routes are chosen less frequently (i.e. worse paths are abandoned).

The architecture of the proposed model

This paper focuses on developing a model for ants route using the information stored locally on the links to decide which next link they should choose. After finding the next link, the *search ant* sends direct message to the source link which is handed over to another ant called *updating ant* that routes through all the visited nests to update routing information stored on them. Meanwhile the *searching ant* can continue the search or die. Each ant class inherits from abstract class provides basic properties used by all ants such as: (i) TTL (Time To Live): This controls the number of hops an ant can make. It is initialized with default TTL value when the

ant is created and then it is decreased by one after each run on the nest. When it reaches zero the ant dies (ii) Run method: The main ant algorithm is implemented in this method that is executed by nest after receiving an ant object (iii) Kill method: This is used by nest when ant has TTL = 0, ant can make some final operations before it is destroyed and (iv) Species: This returns the class name of the ant.

An Ant Search System (ASS) platform proposed as the basic framework on which the algorithm should be implemented is as shown in Figure 2 below. The pheromone applied to edges is an abstraction of the chemical markers used by real ants. Edges with high pheromone levels are more attractive to ants. All ants build their tours using a probabilistic decision rule. The local pheromone update involves decaying the pheromone level on an edge traversed by an ant by a small amount. Once all ants have built a tour, pheromone is deposited along the best ant's tour in a global pheromone update. The whole process then repeats. In this scenario, all *Place ants* register with one another using a service description containing their node's coordinates. *Place agents* can then calculate the distance to any other *Place ant*. *Place ants* maintain a record of the pheromone levels on each link connecting to other Place ants. The ASS platform architecture of the system and the solution agent lifecycle consists of the interactions with the pheromone infrastructure is depicts in Figures 2 and 3 below respectively. The two major types of ants used in this model are *search ant* and *update ant*.

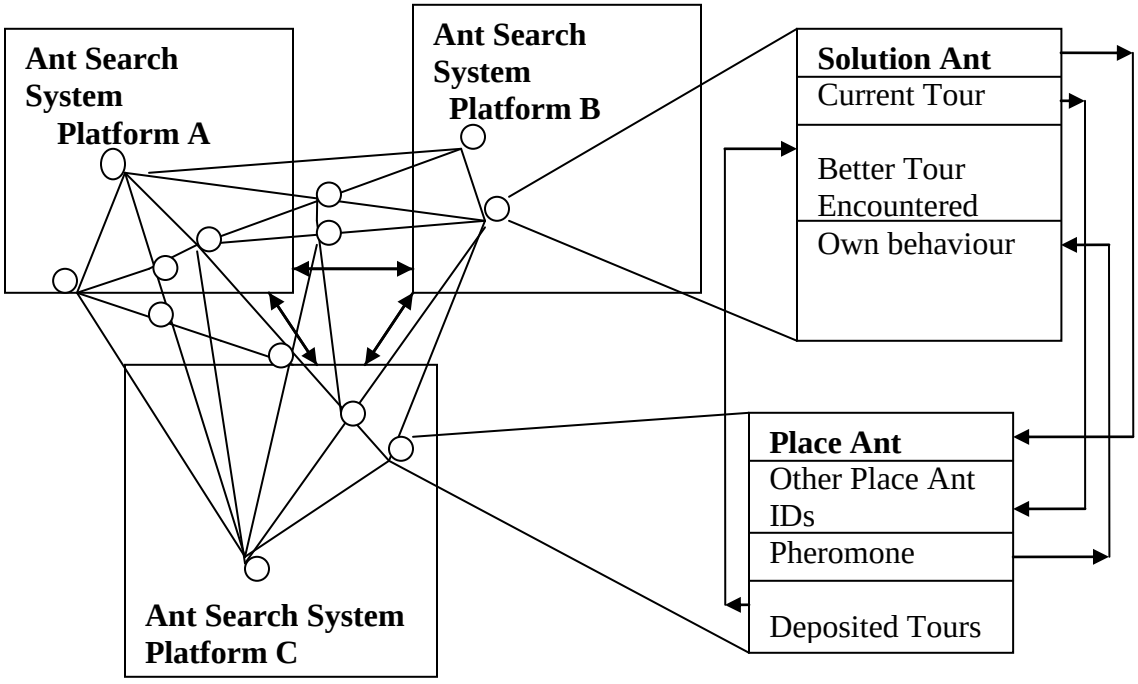

Figure 2: The architecture of the proposed model

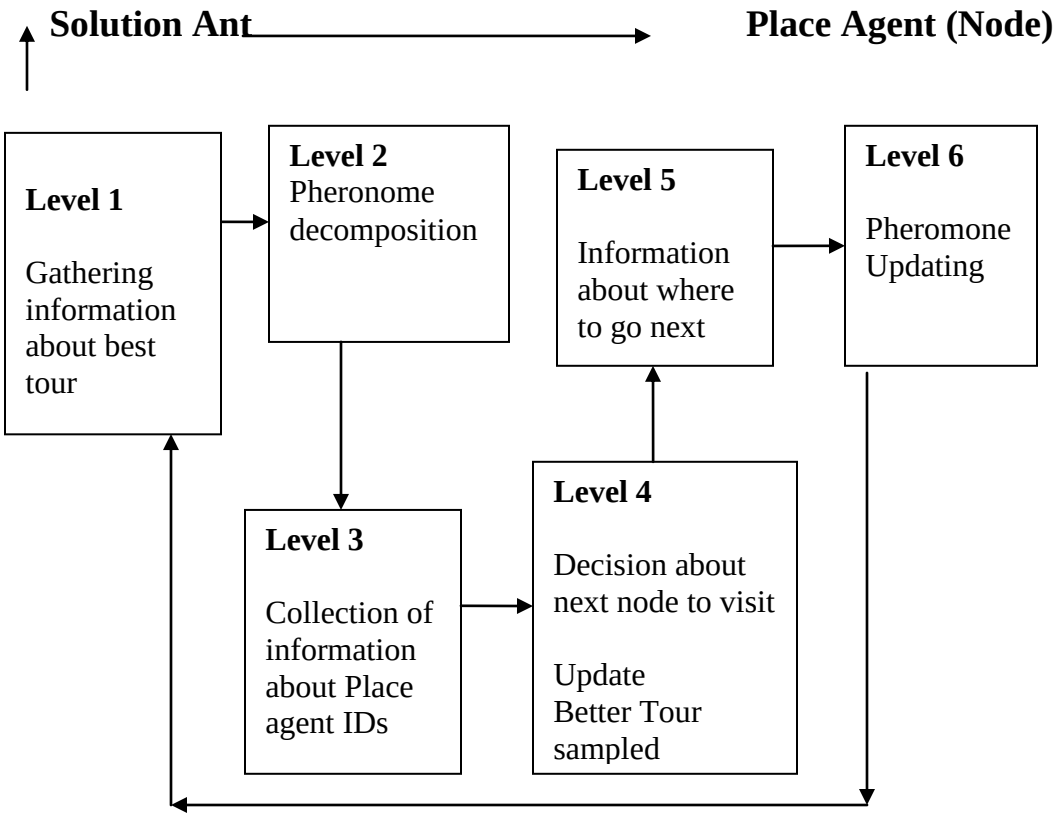

Figure 3: The interaction diagram of the proposed model

Level 1 in Figure 3 above gathered information about best tour and detected the solution agent arrival at a given point with information message containing details of where the solution agent has come from as well as best tour cost. In **Level 2**, the *place ant* performs local pheromone decay on the relevant links. This help to avoid random visit to irrelevant edges. The activities carried out in **Level 3** allows the *place ant* to respond with: (i) a list of other *place ant* identity (acquired from the *Place ant* registrations), (ii) the calculated cost to each of the other *Place ants* and (iii) the latest pheromone level on the links to each of the other *place ants*. In **Level 4,** the *Solution agent* uses available information to decide which node to visit next and informs the *place ant* of its decision to move to a given destination and update better tour previously sampled while information about where to go next was obtained in **Level 5**. In **Level 6**, the *place ant* updates its pheromone value for that link using the equivalent of the algorithm for local pheromone update. The life cycle then returns to Level 1, with the solution agent arriving at the destination. The pseudocode of the model is as depicts in Figure 4 below.

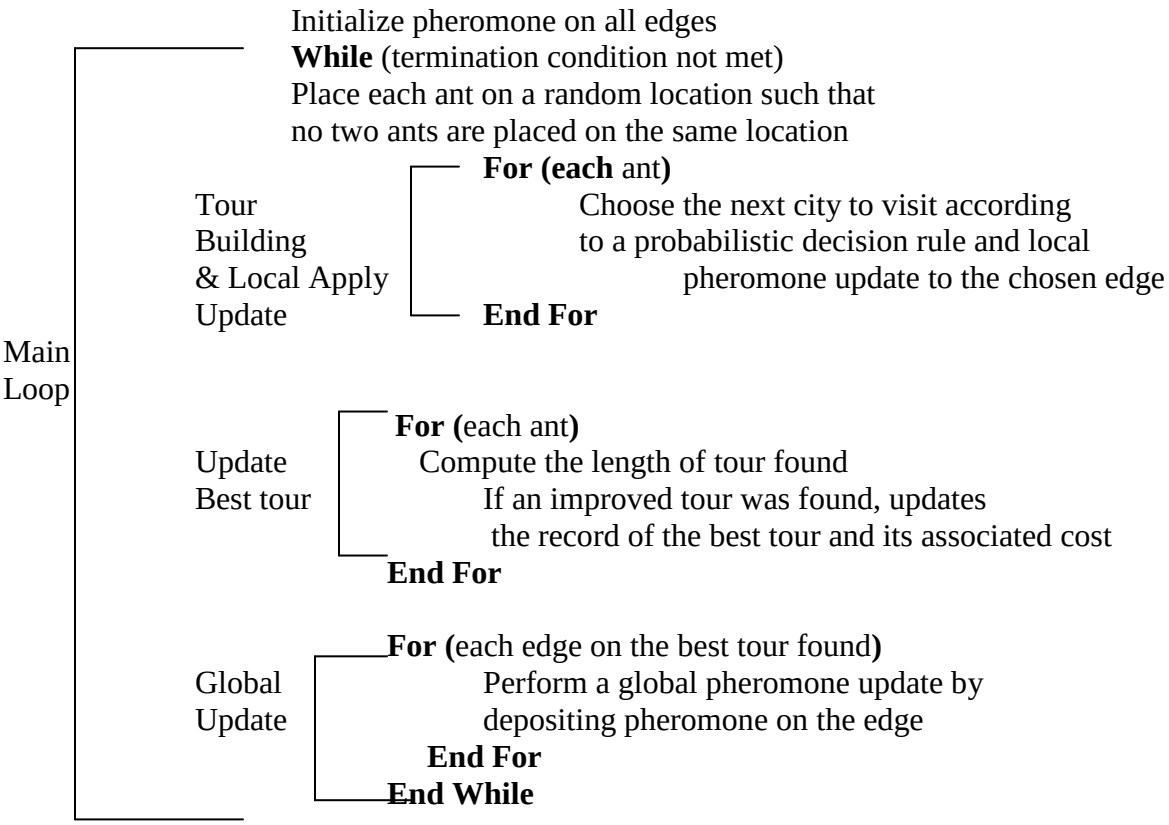

Figure 4: Pseudocode of the model.

Searching Ants and Update Ant

Searching ants are the ants that move through the network of the paths to satisfy users' request. This includes Random Ant, Learning Ant, Hash Ant and Broadcasting Ant classes. Random Ant is the ant that moves between paths randomly. It does not store any data except the description of links that it has to find (inherited from Searching Ant). The Learning Ant routes through the network using knowledge gathered from previous searches. It requires Learning Service that stores results of earlier searches and allows fast search for matching resource description. It is more advanced than Broadcasting and Random ants. Learning Ant remembers its route and all visited locations. The Hash Ant which is the most advanced species of ants. Hash ant uses hash routing service to get the same searches or a set of the closest similar searches, where closest mean searches with the smallest absolute difference between hashed key values.

Broadcasting Ant searches the paths using similar method as in flooding P2P networks. This ant uses Broadcasting Service that is responsible for killing broadcasting ants that visiting the same nest more than once. Updating ant is the ant that visit links to update routing information after a searching ant had found a path. Single Update Ant is used in collaboration with all searching ants that require routing storage updating. Update ant also work with Learning

ants or Hash ants to increase the overall performance of the system. The function of the Update Ant on the system is to updating storage information.

Learning, Hash Routing and Broadcasting Services

The routing storage maintained by Learning service is a dictionary containing the search description and a list of paths that have specified in the search description. Hash Routing Service is similar to Learning Service but additionally it allows searches with the closest hashed key value. Broadcasting Service stores information about ants that visited current path and registered them. This information service stops an ant from broadcasting to all neighbours for the second time.

Mathematical Model of the Proposed System

The mathematical model is formulated based on the following assumptions.

i. The classical traveling salesman problem can be a graph where each city is denoted by a point (or node) and lines are drawn connecting every two nodes (called arcs or edges).
ii. A distance (or cost) is associated with every edge.
iii. If in a graph edges are drawn connecting any two nodes, then the graph is said to be complete.
iv. A round-trip of the cities corresponds to a special subset of the lines when each city is visited exactly once, and it is called a tour.
v. The length of a tour is the sum of the lengths of the lines in the round-trip.
vi. Asymmetric and symmetric TSPs can be distinguished depending on if any edge of the graph is directed or not.
vii. To formulate the symmetric case with n nodes $c_{ij} = c_{ji}$, so a graph can be considered where there is only one arc (undirected) between every two nodes.

Based on the above assumptions, the problem can be solved using the combination of (i) Combinatorial analysis and (ii) Integer Linear programming.

Combinatorial Analysis

The estimation of the number of links to be visited is done using combinatorial analysis. Each link in graph G has n-1 outgoing edges, where n is the number of links. Therefore, the total number of traversable edges by ant is given by:

$$^{n}C_{r} = \frac{n!}{(n-r)!r!} + n$$

This is modified as;

$$^{n}C_{2} = \frac{n!}{(n-2)!2!} + n \text{ ---------- } (1)$$

where n is the number of links that have to be visited for value computation and r = 2 is the distinct paths between them. In equation 1 above the number of nodes can be separated into significant and insignificant edges.

$$\text{Significant Edges} = {}^{n}C_{2} = \frac{n!}{(n-2)!2!} \text{ ---------- } (2)$$

and

Insignificant Edges = n

The shortest path computation path can then be obtained by computing the number of ways of arrangement of nodes can be made using permutation method as follows;

$$^{n}P_{r} = \frac{n!}{(n-r)!} \text{----------(3)}$$

But in this case there is always a starting point and end point. Therefore the arrangement will be given by

$$^{n-2}\mathrm{P}_{r-2} = \frac{(n-2)!}{((n-2)-(r-2))!} \text{----------(4)}$$

Or

$$^{n-2}\mathrm{P}_{r-2} = \frac{(n-2)!}{(n-r)!} \text{----------(5)}$$

This gives the equation for ordering the route in the order in which the nodes are to be visited. After all the node orders are arranged, the one with the least cost in terms of response time is then chosen by ant to traverse its movement round the network. For a symmetrical network there are 1/2 (n-1)! possible tours (because the degree of freedom is (n-1) and tours describing the same sequence but opposite directions are not considered as different tours) and for asymmetric networks where $c_{ij} \neq c_{ji}$ the number of possible tours is (n-1)!.

Integer Linear Programming

This refers to formulating the problem of finding the shortest route for ant journey as an integer programming problem, where the goal is to minimize an objective function for path cost computation subject to a set of constraints. This is also modeled as follows:

Let $x_{ij} = \{0,1\}$ be the decision variable (i= 1,2,…,n and j=1,2,…,n), and $x_{ij} = 1$, means that the arc connecting node i to node j is an element of the tour.

Let $x_{ii} = 0$ (i= 1,2,…,n) ……. *(6)*

meaning that no tour element is allowed from a node to itself. Furthermore

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} \text{ ---------- (7)}$$

that is, the number of decision variables where $x_{ij} = 1$ is equal to *n*, and

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall \in j\{1,2,3,.....,j\} \text{ ……(8)}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall \in i\{1,2,3,.....,i\} \text{ ……(9)}$$

meaning that each column and row of the decision matrix has a single element with a value 1 (i.e., each city is visited once). For assuring the close circuit, an additional constraint must be set. A permutation of nodes *(p1,p2,….pn)* has to be constructed so that the total cost *C(p)* is minimal:

$$\min imiseC(p) = \sum_{i=1}^{n}\left(C_{p_i,p_{i+1}}\right) + C_{p_n,p_1} \text{ ……..(10)}$$

This can be further reduced to

$$\min \sum_{i=j}^{n} \sum_{j=1}^{n} C_{ij} X_{ij} \quad \text{----------} \quad (11)$$

$$s.t \sum_{i=1}^{n} X_{ij}, j = 1,...n \quad \text{----------} \quad (12)$$

$$\sum_{j}^{n} X_{ij}, i = 1,...n \quad \text{----------} \quad (13)$$

$$\sum_{i \in S}^{j} \sum_{j \in S} X_{ij} \leq |S| - 1 \quad \text{----------} \quad (14)$$

Integer variables X_{ij} indicates whether or not the node i is to be visited after a node j, respectively with (X_{ij}=1) or (X_{ij}=0). The costs of going from node i to node j are given by C_{ij} where n is the number of nodes in the problem, S is a nonempty, proper subset of the set {1,2,3,4,5,…, n}. Equation 11 is the problem general representation, indicating that the minimization of the summation of the path costs among all connections between the different nodes is being sought. Equations 12,13 and 14 represent the restrictions that have to be respected. Moreso, equations 12 and 13 guarantee that every node must have a connection coming from one node and another going to another node while the inequality 14 guarantees that no subpath can be created. This equation can then be solved to obtain the path with minimum cost (shortest path).

Path Computation Algorithm of the proposed model

The path computation algorithm of the model is capable of selecting the actual shortest path rather than the best effort result by combination of different ant as agent to search for the shortest path in the search space through some numbers of iteration and use of pheromone update information. The algorithm for the model is as follows:

Initialize all the ants with initial pheromone value
Start searching for optimum value in the search space from a random position
Update the pheromone values of all the ants
Compute the solution for each ant
Locally Search for the optimum value
Compare the Local Minimum with the Global Minimum
IF Local_Minimum > Global_Minimum
Reject
End IF
IF Global_Minimum= Local_Minimum
Update the pheromone of the ant, which generates the Local Minimum
Repeat for N times
End IF

The above main procedure has three components: (i) management of the ants' activity, (ii) pheromone updating, and (iii) decision making actions. However, the above procedure

require optimization which can be achieved through the formulated mathematical models in equations 1, 2 and 3 above. The algorithm for this is as follows:

Run(nest):
Add nest to ant's route list
Remove cycle from ant's route list
Add path to the visited paths set
Get Learning Service from path
If Learning Service is not present **then**:
Move to random neighbour of path
else:
End If
If resources found on path:
Send UpdateAnt and inform source node
If random < RandomSearchProbability **then**:
Move to random neighbour path not visited
else:
End if
End if
Procedure (Ant Colony Optimization)
Initialize all the ants with initial pheromone value ($T_0 = 0.001$).
Accumulate pheromone trails, ρ , calculate heuristic information
While (termination condition not met) **do**
Construct Solutions (pheromone trails, ρ , heuristic information)
$T_{new} \leftarrow (1 - \rho) * T_{old} + \rho * T_0$
(where T_{new} is the updated pheromone value, T_{old} is the old pheromone value)
LocalSearch(p); optional
GlobalUpdateTrails(p);
minimum: $T_{new} \leftarrow (1 - \alpha) * T_{old} + \alpha * \Delta T_{old}$, where $\alpha \leq 1$ is a constant value.
Update the pheromone for ants that generate local minimum:
$T_{new} \leftarrow (1 - \alpha) * T_{old}$, where $\alpha \leq 1$ is a constant value.
End do
Setting the Link Weights
Initialize
concatenate (all links)
Path =CONTE(links)
link wt. = sum of all links from source to destination
Td = Sum intersec(CONTE(links))
While (termination condition not met) **do**
Shortest path = Min(link wt)
ReturnMin(link wt)
End do
End Ant Colony Optimization

Performance evaluation and results

With real world, order of connection set up in shortest path algorithm matters. This is demonstrated in Figure 5 below.

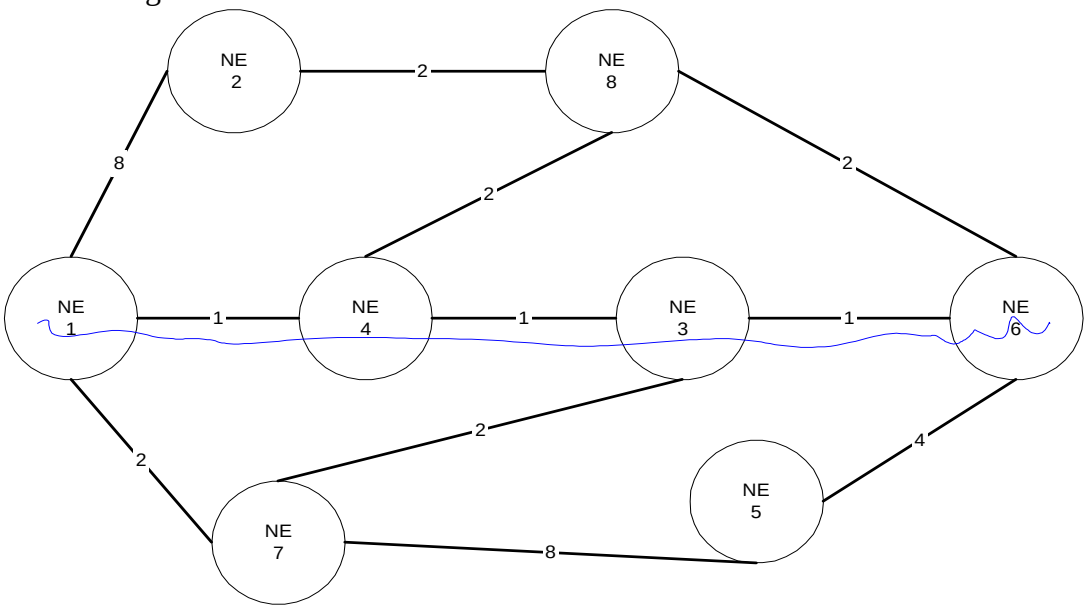

Figure 5: All possible links between N1 (source) and N6 (destination)

By using analytical solution and considering all possible paths between nodes NE1 and NE6 with NE1 as the source and NE6 as the destination, the following are the possible links.

$NE1 \rightarrow NE4 \rightarrow NE3 \rightarrow NE6 = 3;$
$NE1 \rightarrow NE2 \rightarrow NE8 \rightarrow NE4 \rightarrow NE3 \rightarrow NE6 = 14;$
$NE1 \rightarrow NE2 \rightarrow NE8 \rightarrow NE6 = 12;$
$NE1 \rightarrow NE7 \rightarrow NE3 \rightarrow NE6 = 5;$
$NE1 \rightarrow NE7 \rightarrow NE5 \rightarrow NE6 = 14$

The shortest route is $NE1 \rightarrow NE4 \rightarrow NE3 \rightarrow NE6 = 3$; though this route contains four nodes like other three routes with values 12, 5, and 14 but these routes returns higher values. This indicates that none of them is the best solution i.e. not the route to be considered. Also, two routes return the same value, 14, but one is with five nodes while the other is with four nodes. From this, it has been observed that less number of nodes to be transversed is not necessarily means the shortest and best route. The same analysis can be done for other paths; NE2 to NE6; NE3 to NE6; NE4 to NE6; NE5 to NE6; NE7 to NE6 and NE8 to NE6 where NE2, NE3, NE4, NE5, N7 and N8 can be considered as the sources and NE6 is the destination.

The formulated model was implemented using C# programming language. All possible routes that link all nodes in figure 5 above with N1,N2,N3,N4,N5,N7 and N8 as sources and N6 as the destination are generated as shown in Table 1 below. All possible routes with shortest route in each case were generated as indicated in Tables 2 to 8 and the response graph for each result are shown in Figures 6 to 12.

Table 1: Input data for possible routes

	N1	N2	N3	N4	N5	N6	N7	N8
N1	0	8	0	1	0	0	2	0
N2	8	0	0	0	0	0	0	2
N3	0	0	0	1	0	1	2	0
N4	1	0	1	0	0	0	0	2
N5	0	0	0	0	0	4	8	0
N6	0	0	1	0	4	0	0	2
N7	2	0	2	0	8	0	0	0
N8	0	2	0	2	0	2	0	0

Table 2: Simulation output for possible routes between (N1 to N6)

Route	Path	Distance
Route1, (R1)	1,4,3,6	3
Route2, (R2)	1,2,8,4,3,6	14
Route3, (R3)	1,2,8,6	12
Route4, (R4)	1,4,8,6	5
Route5, (R5)	1,7,5,6	14

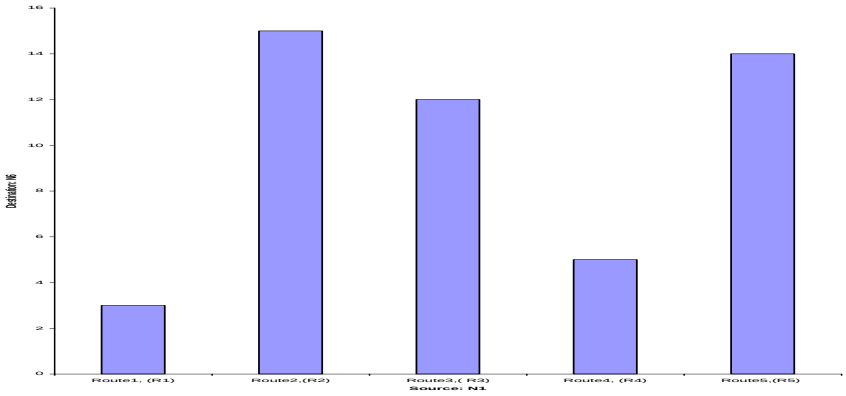

Figure 6: Result for N1 (source) to N6 (destination)

Table 3: Simulation output for possible routes between (N2 to N6)

Route	Path	Distance
Route1, (R1)	2,8,6	4
Route2, (R2)	2,8,4,3,6	6
Route3, (R3)	2,1,7,5,6	22
Route4, (R4)	2,1,4,3,6	11
Route5, (R5)	2,1,7,3,6	13

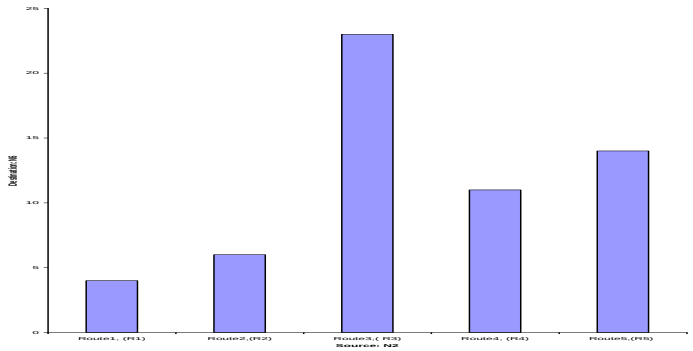

Figure 7: Result for N2 (source) to N6 (destination)

Table 4: Simulation output for possible routes between (N3 to N6)

Route	Path	Distance
Route1, (R1)	3,6	1
Route2, (R2)	3,4,8,6	5
Route3, (R3)	3,7,5,6	14
Route4, (R4)	3,7,1,2,8,6	16
Route5, (R5)	3,4,1,2,8,6	14

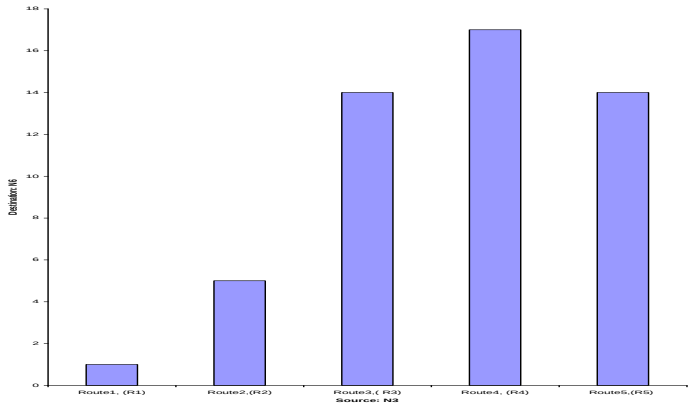

Figure 8: Result for N3 (source) to N6 (destination)

Table 5: Simulation output for possible routes between (N4 to N6)

Route	Path	Distance
Route1, (R1)	4,3,6	2
Route2, (R2)	4,1,2,8,6	13
Route3, (R3)	4,8,6	4
Route4, (R4)	4,1,7,3,6	6
Route5, (R5)	4,1,7,5,6	15

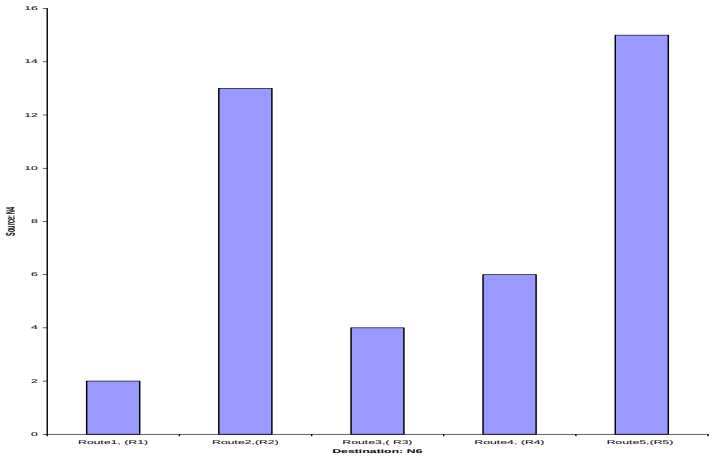

Figure 9: Result for N4 (source) to N6 (destination)

Table 6: Simulation output for possible routes between (N5 to N6)

Route	Path	Distance
Route1, (R1)	5,6	4
Route2, (R2)	5,7,3,6	11
Route3, (R3)	5,7,1,2,8,6	22
Route4, (R4)	5,7,1,4,3,6	13
Route5, (R5)	5,7,1,2,8,4,3,6	24
Route6, (R6)	5,7,1,4,8,6	15

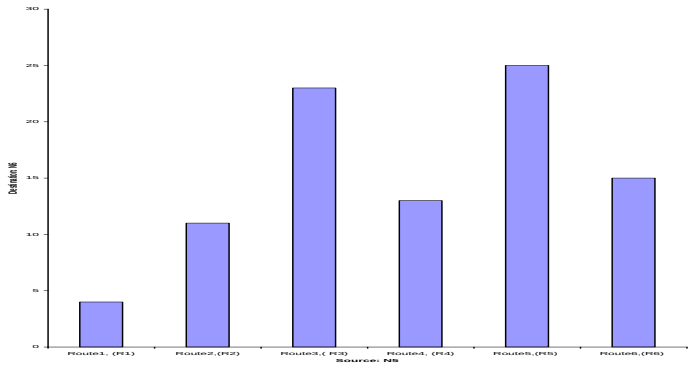

Figure 10: Result for N5 (source) to N6 (destination)

Table 7: Simulation output for possible routes between (N7 to N6)

Route	Path	Distance
Route1, (R1)	7,5,6	12
Route2, (R2)	7,3,6	3
Route3, (R3)	7,1,4,3,6	5
Route4, (R4)	7,1,2,8,6	14
Route5, (R5)	7,1,2,8,4,3,6	16

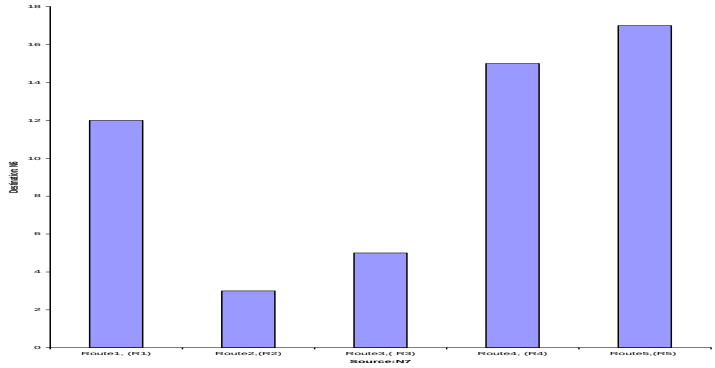

Figure 11: Result for N7 (source) to N6 (destination)

Table 8: Simulation output for all possible routes between (N8 to N6)

Route	Path	Distance
Route1, (R1)	8,6	2
Route2, (R2)	8,4,3,6	4
Route3, (R3)	8,4,3,7,5,6	17
Route4, (R4)	8,2,1,4,3,6	13
Route5, (R5)	8,2,1,7,5,6	24
Route6, (R6)	8,2,1,7,3,6	15

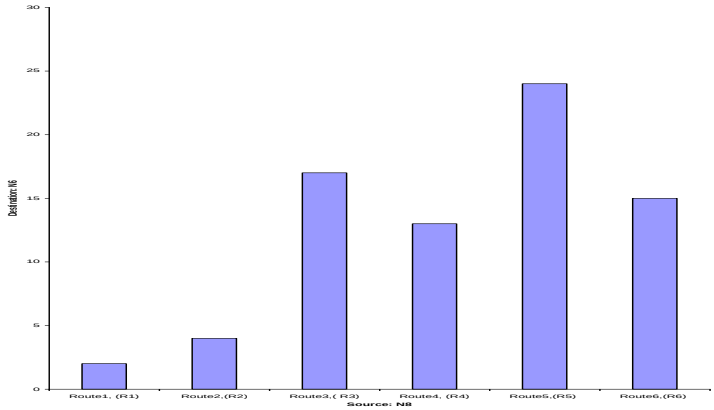

Figure 12: Result for N8 (source) to N6 (destination)

Contribution to Knowledge

The proposed model introduced a new variant of shortest route solution in which the input of a link towards the route length depends not only on the weight of the link itself but also on the weight of the links transverses before and after transeversing the link under consideration. The main advantage of this decomposition approach is that it permits the use of mixed integer programming and combinatorial optimization technique to compute proven optimal routing path, solving the problem in practice by identifying and generating all possible solution space (routes) with their distance values and identify the actual shortest route among the possible solution space (routes).

Conclusion

In this paper, mathematical modeling solution is proposed using Ant Colony Search Algorithm (ACSA) approach for the determination of global optimum solution for TSP and tested on ASS platform architecture. The performance evaluation of the model demonstrated that it is capable of undertaking global search and identify all possible solutions (routes) and identify the best solution (shortest route). From the simulation using Mathlab 7.0 and implementation using C# programming language, it has been observed that our model's solution is better than the analytical solution because the analytical solution and models in Applegate et al. (2006), Shi et al.(2007), Bontoux and Feillet (2008), Yu-Wan et al (2007) and Dorigo et al (1996) are just *best effort* solutions and not intelligent to identify and point out the best solution (shortest route) to the users. However, our model can be further improved upon to achieve better solution for resource location and discovery on peer-to-peer network system as well as other similar problems such as Minimum Spanning Tree Problem (MST),Vehicle Routing, Sequential Ordering, Connection-oriented Network Routing, Connectionless Network Routing, Optical Network Routing, Shortest Common Super sequence, Constraint Satisfaction, 2D-HP Protein folding and other related problems.

Recommendation

Complex systems like Traveling Salesman Problem (TSP) have interesting properties like; total decentralization, tolerance to changes and self-organization to achieve global stability, Alberto et al.(2001). Generally these systems have a set of properties. Such properties as highlighted in Francesco (2002**)** are: *(i) Positive feedback*, *(ii) Negative feedback* and *(iii) Fluctuations.* Ant Colony Search (ACS) systems should therefore have all of the properties highlighted above to achieve self organization, adaptability and seeking for the best solution. The above properties can be used to further improve on this model to accomplish better result.

References

Alberto Montresor, Ozalp Babaoglu, Hein Meling. (2001): Anthill: a Framework for the Design and Analysis of Peer-to-Peer Systems. Department of Computer Science, University of Bologna.

Ammar,E.E., Youness, E.A. (2005): Study on multiobjective transportation problem with fuzzy numbers, *Applied Mathematics and Computation 166 (2005) pp. 241-253.*

Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J. (2006): The Traveling Salesman Problem, A Computational Study, Princeton University Press, Princeton and Oxford pp.10-11.

Bontoux, B., Feillet, D. (2008): Ant colony optimization for the traveling purchaser problem, *Computers & Operations Research 35 (2008) pp.628-637.*

Brueckner, S. (2002): Return from the Ant: Synthetic Ecosystem for Manufacturing Control, PhD. Thesis, Humboldt University.

Ding, C., Cheng, Y., He, M. (2007): Two-Level Genetic algorithm for Clustered Traveling Salesman Problem with Application in Large-Scale TSPs, *Tsinghua Science and Technology, Vol.12.No.4 pp. 459-465.*

Dorigo, M., V. Maniezzo and A. Colorni, (1996): The Ant System; Optimization by a colony of cooperating agents, *IEEE Transaction on Systems, Man and Cybernetics – Part B, Vol. 26, No.1 pp 1-13.*

Dorigo Marco, (1992): Optimization, Learning and Natural Algorithms. PhD thesis, Politecnico di Milano, Italy.

Francesco Russo, (2002**):** JXTAnthill. Master's Thesis. Department of Computer Science, University of Bologna.

Kikuchi, S., Chakroborty, P. (2006): Place of possibility theory in transportation analysis, *Transportation Research Part B 40 (2006) pp.595-615.*

Lenin, K., M. R. Mohan,(2006): Ant Colony Search Algorithm for Optimal Reactive Power Optimization, *Serbian Journal of Electrical Engineering, Volume 3, No. 1, June 2006, pp 77- 88.*

Parunak, H. V. D, P. Weinstein, P. Chiusano and S. Brueckner, (2004): Agent Swarming in Semantic Space to Corroboratye Hypothesies. In Proceeding of Third International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp 1488-1489.

Ridge, E., D. Kudenko, D. Kazakov and E. Crry, (2005): Moving Nature-Insipired algorithms to Parallel, Asynchronous and Decentralised Environments. In Self-Organization and Autonomic Informatics, pp 35-49.

Shi, X.C., Liang, Y.C., Lee, H.P., Lu, C., Wang, Q.X. (2007): Particle swarm optimization-based algorithms for TSP and generalized TSP, *Information Processing Letters 103 (2007) pp. 169-176.*

Yu-Wan Chen, Yong-Zai Lu, Penf Chen, (2007): Optimization with extremal dynamics for the traveling salesman problem, *Physica A 385 (2007) pp. 115-123.*

This academic article was published by The International Institute for Science, Technology and Education (IISTE). The IISTE is a pioneer in the Open Access Publishing service based in the U.S. and Europe. The aim of the institute is Accelerating Global Knowledge Sharing.

More information about the publisher can be found in the IISTE's homepage: http://www.iiste.org

CALL FOR JOURNAL PAPERS

The IISTE is currently hosting more than 30 peer-reviewed academic journals and collaborating with academic institutions around the world. There's no deadline for submission. **Prospective authors of IISTE journals can find the submission instruction on the following page:** http://www.iiste.org/journals/ The IISTE editorial team promises to the review and publish all the qualified submissions in a **fast** manner. All the journals articles are available online to the readers all over the world without financial, legal, or technical barriers other than those inseparable from gaining access to the internet itself. Printed version of the journals is also available upon request of readers and authors.

MORE RESOURCES

Book publication information: http://www.iiste.org/book/

Recent conferences: http://www.iiste.org/conference/

IISTE Knowledge Sharing Partners

EBSCO, Index Copernicus, Ulrich's Periodicals Directory, JournalTOCS, PKP Open Archives Harvester, Bielefeld Academic Search Engine, Elektronische Zeitschriftenbibliothek EZB, Open J-Gate, OCLC WorldCat, Universe Digtial Library , NewJour, Google Scholar