# Analysis of Ant Colony Optimization Algorithm solutions for Travelling Salesman Problem

**Asma Salem, Azzam Sleit**
**The University of Jordan, Amman, Jordan**
**King Abdullah II School of Information Technology**
**Asma_salem85@yahoo.com, azzam.sleit@ju.edu.jo**

***Abstract*— Ant Colony Optimization is a metaheuristic algorithm used for solving complex combinatorial optimization problems. With inspiring the algorithm implementation by the biological behavior of ants, multiple solutions were proposed in literature to provide solutions for many problems. Based on the behavior of real ants, Ant Colony Optimization algorithm represented good results to several well-known complex problems, such as the travelling salesman problem. The main objective of travelling salesman problem is established to find the shortest visit through a given number of locations, taking into consideration that every city is visited only once. In this paper, several implementations for optimization algorithm are examined and analyzed. These techniques are applied to travelling salesman problem, based on tuning multiple parameters, in which they are affecting the cost of the algorithm and maximizing the performance to reach the best path. It was found that the cost increases with both Number of visited cities and evaporation rate, while it decreases with both number of ants and number of iterations.**

***Keywords*— ant colony; optimization; travel salesman problem; metaheuristic algorithm**

## I. INTRODUCTION

Dorigo in the 1990s [1], suggested that ant colony optimization (ACO) is a metaheuristic algorithm based on swarm intelligence (SI). It is inspired by swarm's behavior, as it is composed of many individuals, who are all homogenous, in which all local communication based on simple rules and well self-organized mechanisms [1, 2]. Followed by Dorigo's research, many researchers start using "artificial ants" to solve difficult combinatorial problems. Based on the behavior of real ants, ACO represented good results for several well-known complex problems, specially scheduling problems and vehicle routing problems [2, 3].

Inspiring source of ACO is based on ants' behavior in seeking for food, by following behavior of real ants, which used chemical pheromones as a communication medium [1]. The colony consists of simple creations, called (artificial) ants, using a pheromone for communication as they mark their previous paths. The pheromone is used in ACO to serve ant as medium for communicating distributed, which the ants use to probabilistically construct paths for food [4].

In the real life scenarios problem which is being solved in similar mechanism; the ants adapt their paths trails during the seeking for food to reflect their search experience for food to attract another ants [5].

ACO has been used in several classes of problems, such as vehicle routing [3], examination of scheduling problem and scheduling tasks [2, 6]. The meaning of an ant's move is highly dependent on the graph representation of the problem. For example, ACO algorithm was used for solving the Traveling Salesman Problem (TSP) [3]. The main objective of TSP is to find the shortest visit through a given number of locations, taking into consideration that every site is visited only once [5]. It is solved by representing each node of the graph as a city and each path has a weight corresponding to the distance between the cities. When the ant moves from one location to another, it means that it travel between two corresponding cities. TSP is a standard problem of combinational optimization of procedures research area. There are several implementations for this problem such as vehicle routing, clustering data arrays and many others problems [4-6]

In this work we represent the mathematical background for ACO optimization algorithm; examining the algorithm optimization solutions proposed for solving problems such as TSP. Several implementations for optimization algorithm are examined and analyzed. These techniques are applied to TSP based on tuning multiple parameters, in which they are affecting the. These implementations of the algorithm were carried on matlab environment.

## II. BACKGROUND

The notion behind ACO is based on the ''natural'' algorithm used by real ants to generate a near-best path between their nest and the food source. It is known that ants are blind, deaf, and almost dump. While their behavior directed to the survival of the colony [3, 4]. During their seeking food process, ants deposit chemical substances called pheromones on their way back to their nest that form atrial for other ants. Other ants sense the chemical pheromone, and the paths become with highly probability to be visited next time. The ants in the next rounds become interested to the marked paths; the more pheromone that is released on a path, the more attractive

that path becomes. This marked trial attracts other ants on next seeking food process [5, 6].

The pheromone is subjected to be evaporated over time. This evaporation rate will be more significant along longer paths. So, shorter paths are refreshed more rapidly, and being more likely to be visited in next seeking food process; therefore having the chance of being more frequently explored by other ants [1, 3].

This simple scheme for ants seeking food is described in figure (1). Initially, ants tend to seek food randomly; in the left part, as they start to mark their paths using pheromone; then the path would be determined and with highly chance to be visited by the next seekers as shown in right part on figure (1). Naturally, ants will visit the marked paths and tend towards the most efficient path due to the fact that it characterized by the highest density of chemical pheromone, which is the main communication medium being used by ants [4, 5].

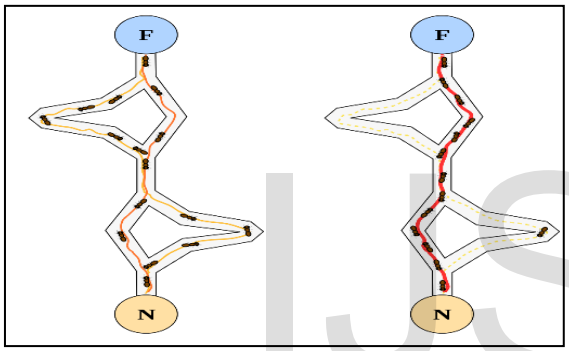


**Figure (1): Ants seeking for food using pheromone communication scheme**

The main goal of TSP is finding the shortest tour through a set of cities starting from one city and going through all other cities once and only once, returning to the starting city [6-9]. TSP also means finding the shortest Hamiltonian path in a fully connected graph; this implies that TSP belongs to the class of non-deterministic polynomial-time hard problems (NP). Based on the solutions afforded in literature there is no efficient algorithm for solving TSP. The goal is to minimize the total distance length (reach best path in term of distance) [12, 13, 14].

Dorigo create computational agents called ants and mimic the behavior of real ants going from their nests to a food source. To do so, the “artificial ants” (called just “ants”) are placed on a graph and forced to move towards food in different paths. Each single movement of the ant on the graph generates another piece of the solution. The set of moves generated the full problem solution [5, 6].

The statement of the problem in TSP is simple; it remains one of the most challenging hard problems for computer researchers; because it is difficult to find the shortest closed tour that connects all the given cities by the given tour. TSP is consisted of a list of cities and the task is to find a shortest possible tour that visits each location/city exactly once [7, 10].

As shown in figure(2) and based on the ACO algorithm behavioral, the TSP could be identified the salesman workers as the ants working in the colony, the cities are the node of food, and the paths for the food identifying the route of salesman visits for each city. Finding the shortest path for the cities visits could be inspired by ACO algorithm behavior [9, 10, 11].

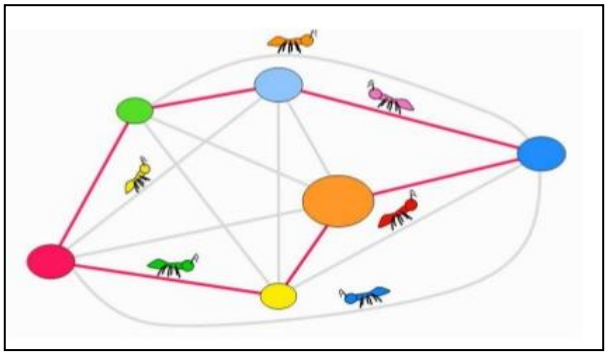

**Figure (2): ACO algorithm provide solution for the TSP**

## III. ACO algorithm and TSP

Ant Colony algorithms are usually used to solve the minimum cost problem [11, 12]. This work will discuss the solutions afforded to the TSP using ACO optimization algorithm. The two main phases of the ACO algorithm are the ants' route construction and pheromone update, they leads to find the shortest path and generate the minimum cost to solve the problem [14-16]. The algorithmic steps are shown in table (1). This algorithm can be explained as follows [1]:

1) Each iteration begins with the positioning of the ant on a starting node following a specific transition rule.
2) All ants are then moved according to a transition rule, until they create a complete solution (a solution is a set of moves and will represent the shortest distance) [2].
3) To choose the next node, the transition rule is being chosen based on the probability of an ant choose a path. This probability is normally calculated in each step to determine the next node will be chosen to be visited [9]. ACO algorithm represented a good solution for TSP in

the literature; this algorithm flowchart is described in figure (3).

**Table (1): The ACO pseudo-code [1]**

**Algorithm 1**
**1 Initialize**
**2 Repeat (at this level, each execution is named iteration)**
**3 Each ant is positioned on the initial node**
**4 Repeat (at this level, each execution is named step)**
**5 Each ant uses a transition rule to increment the solution set**
**6 Update the pheromones according the local pheromone update rule**
**7 Until all ants have built a complete solution**
**8 Apply a local search procedure**
**9 Apply a pheromone global update rule**
**10 Until a stop criteria is satisfied**

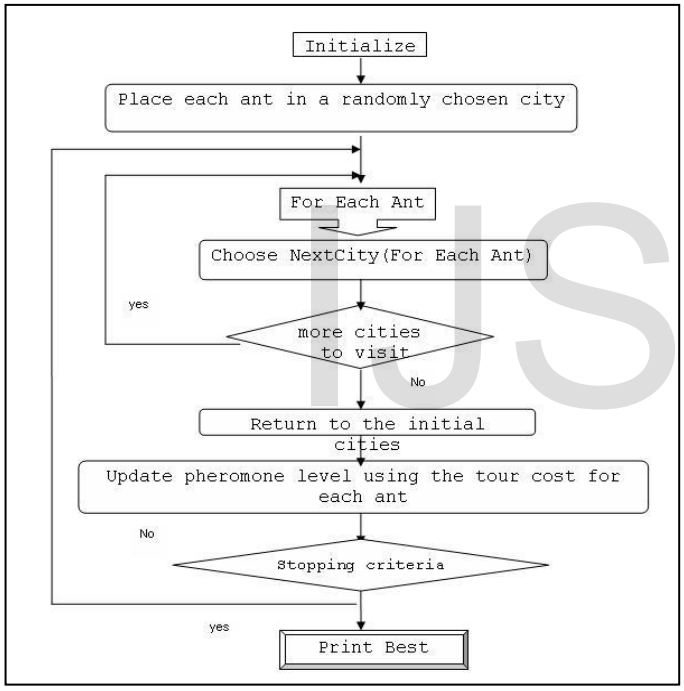


**Figure (3): ACO algorithm solved the TSP**

The ants are representing workers (salesman) in the colony [3]. There are two working modes for the ants; the forwards or backwards. The ant memory allows them to retrace the path which was followed while searching for the next desired node [9]. By Keep moving backward on their memorized paths, ants leave pheromone on the paths they traversed. The best path is constructed using two main phases in the previous code of ACO, the path construction and the pheromone update [3]; in the path construction phase, number of ants concurrently building their tours beginning from starting nodes which were selected randomly in the network of number of nodes (represent cities in TSP). At each construction step, any ant say ant (k) currently at selected node, say node (i) applies a probabilistic random proportional rule to decide which node to go next [4, 6]. The ants select the move to expand its path by taking into account the following two values: Heuristic function (ηij) and Level of pheromone ( $\tau_{ij}(t)$ ) described in table (2) and equation (1):

**Table (2): ACO important parameters [1, 3]**

| Values | Description |
|---|---|
| Heuristic function $(n_{ij})$ | Represents the preferable path for the ant to move at each step<br>Calculated as the inverse of the distance/cost on the path from node i to node j |
| Level of pheromone $\tau_{ij}(t)$ | Level of pheromone on the path between nodes (i,j) |

Given these parameters, the probability with which the ant chooses to go to node (n) in the next tour :

$$p_{ij}(t) = \frac{\tau_{ij}(t)^{\alpha}\left(n_{ij}\right)^{\beta}}{\sum_{l \in N^k} \tau_{il}(t)^{\alpha}\left(n_{il}\right)^{\beta}} \qquad (1)$$

**If node (n) belongs to $N_i^K$ . And $p_{ij}(t)$ = 0, otherwise.**

Where, $N_i^K$ is the adjacent neighborhood, (α) and (β) are heuristic parameters. Each ant maintains a memory of the nodes already have been visited in previous iterations [3]. Once all ants have completed a tour, the pheromone trails are updated. This is done by first lowering the pheromone levels on all paths benefits of evaporation process; in order to force the running of the algorithm to forget bad solutions and encourage exploration of new paths) and then adding pheromone to the paths that have been traversed[2, 9]. The Ant Colony Optimization algorithm has the following main functions [1, 3]:

a) ***Route Construction***: During the path construction, ant k, located at node (i), moves to node (n) chosen according to the following pseudorandom proportional rule:

A random variable say (q) which is uniformly distributed over [0; 1], and if q > $q_0$ the node (n) is chosen according to the equation (1), using α = 1. Otherwise, choose n by equation (2) below.

$$\mathbf{N = arg\ max}_{\ j \in} {}^{N_i^K} \{ (\eta_{ij})( T_{ij} )^{\beta} \} \qquad (2)$$

With probability $q_0$, the ant makes the best move described by the pheromone trails and heuristic information (investing the learned knowledge), and with probability = (1- $q_0$) it performs a biased discovery of the paths [3, 5, 6].

b) ***Pheromone Update***: The ants uses two types of pheromone updates to maintain the paths knowledge which being learned and memorized from the past tours; global and local in which they are described in table (3):

**Table (3): ACO Pheromone updates types [1, 3]**

| Type | Local update | Global update |
|---|---|---|
| **Description** | **performed every time an ant traverses an path between nodes (i,j)** | **It is only carried out by the ant that finds the best tour so far. It enables the algorithm to converge faster by concentrate the search around the best path.** |
| **Equation** | $\tau_{ij} \leftarrow (1-\xi)\tau_{ij} + \xi\tau_0$, **(3)** | $\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \quad \forall(i,j) \in T^{bs}$ **(4)** |
| **Parameters** | **Where $0 < \xi < 1$ $\tau_0$ is the initial pheromone $\tau_0 = (NL^{nn})^{-1}$ where $L^{nn}$ is the length of the nearest neighbor tour (nearest unvisited node).** | **Where delta $T^{bs}_{ij} = (L^{bs})^{-1}$ ρ, is a parameter governing decay: $0 < \rho < 1$ $T^{bs}$ is the best found tour so far with $L^{bs}$ its length.** |

The pheromone deposit is updated into two main strategies:

a) A positive reinforcement strategy of the algorithm and it tends to provide the best solutions [1, 3].

b) A negative reinforcement strategy, or called pheromone evaporation, formulated as a multiplication of all pheromone values by a heuristic coefficient ($0 < \rho < 1$).

After multiple iterations, the best solution will tend to eliminate the poor ones, allowing all the ants to choose one single route and declared the best path at the end of algorithm execution [9, 10].

## IV. Implementation of ACO in Matlab environment

In this section matlab environment was used to execute the algorithm of ACO in different implementation and multiple tuning parameters. The code and implementation of ACO was published on the math work website [15]. It was freely downloaded and being used with different values for different experiments to evaluate the performance of ACO optimization algorithm.

The full source code was downloaded and being implemented with different parameters [15]. This work highlighted the main factors affecting the algorithm performance. Multiple parameters are being tuned to reflect the optimization factor for the algorithm; the parameters are described as follow:

a) The number of worker in the colony affecting the overall cost of the algorithm. This part described the effect of changing the ant's number in each execution, and collecting the results for cost (distance); the cost reduction was highlighted in this experiment, when the increase of the workers in the colony happened (the number of ants in each execution). Table (4) and figure (4) described the results.

**Table (4): Experiment 1.a parameters and results**

| %% ACO Parameters<br>Number of Iterations =1000<br>Number of Ants =changed<br>Number of cities=1000<br>Evaporation Rate=0.4817 | **Number of Ants** | **Cost (Distance)** |
|---|---|---|
| | **20** | **531** |
| | **40** | **522** |
| | **80** | **514** |
| | **160** | **512** |

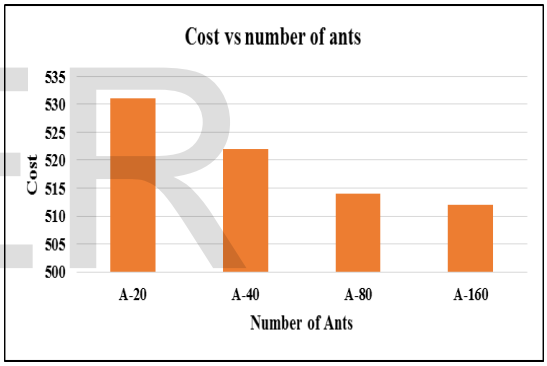


**Figure (4): Experiment 1.a results**

b) The number of cities in the colony affecting the overall cost of the algorithm. This part described the effect of Changing the cities number in each execution, and collecting the results of cost; the cost growth is highlighted in this experiment; as the algorithm will take less cost (less trip length) if the number of cities was reduced. Table (5) and figure (5) described the results.

**Table (5): Experiment 1.b parameters and results**

| %% ACO Parameters<br>Number of Iterations =1000<br>Number of Ants =20<br>Number of cities=changed<br>Evaporation Rate=0.4817 | **Number of cities** | **Cost (Distance)** |
|---|---|---|
| | **250** | **130** |
| | **500** | **268** |
| | **1000** | **525** |

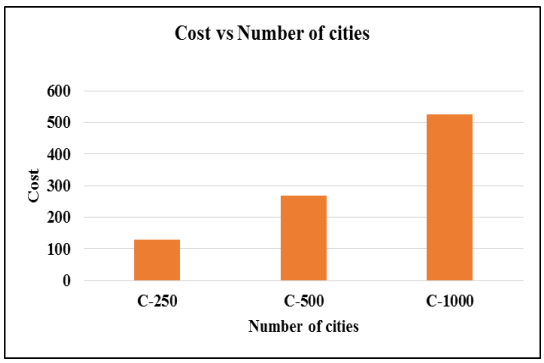


**Figure (5): Experiment 1.b results**

c) The number of iterations in which the ant will conduct to solve the problem affecting the overall cost of the algorithm. This part described the effect of changing the iterations number in each execution, and collecting the results of cost; the cost reduction is highlighted in this experiment; as the algorithm will take less cost (less trip length) if the chance of the algorithm will increase to be solved (by increasing the iterations). Table (6) and figure (6) described the results.

| %% ACO Parameters<br>Number of Iterations =changed<br>Number of Ants =20<br>Number of cities=1000<br>Evaporation Rate=0.4817 | **Number of iteration** | **Cost (Distance)** |
|---|---|---|
| | **50** | **530** |
| | **250** | **521** |
| | **500** | **514** |
| | **1000** | **512** |

**Table (6): Experiment 1.c parameters and results**

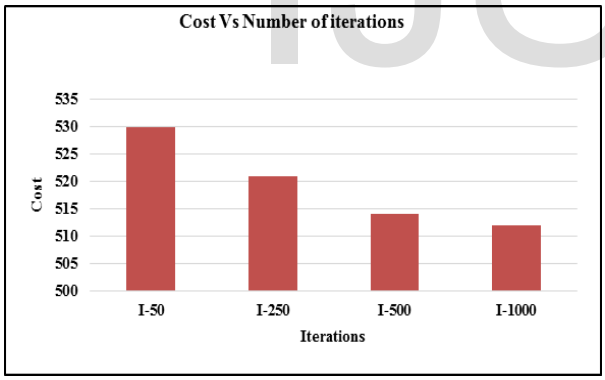


**Figure (6): Experiment 1.c results**

d) The evaporation rate of the pheromone in which the ant will used to marked up the trip paths for each city affecting the overall cost of the algorithm. This part described the effect of changing the evaporation rate in each execution and collecting the results of cost; the cost reduction is highlighted in this experiment; as the algorithm will take slower rates in updating the pheromone values in each iteration carried by the algorithm. The increases in the evaporation rate value will surely maximize the algorithm cost. Table (7) and figure (7) described the results.

| %% ACO Parameters<br>Number of Iterations =1000<br>Number of Ants =20<br>Number of cities=1000<br>Evaporation Rate=changed | **Evaporation rate** | **Cost** |
|---|---|---|
| | **0.00005** | **509** |
| | **0.0005** | **518** |
| | **0.005** | **520** |
| | **0.05** | **522** |
| | **0.125** | **526** |
| | **0.25** | **528** |
| | **0.4817** | **534** |

**Table (7): Experiment 1.d parameters and results**

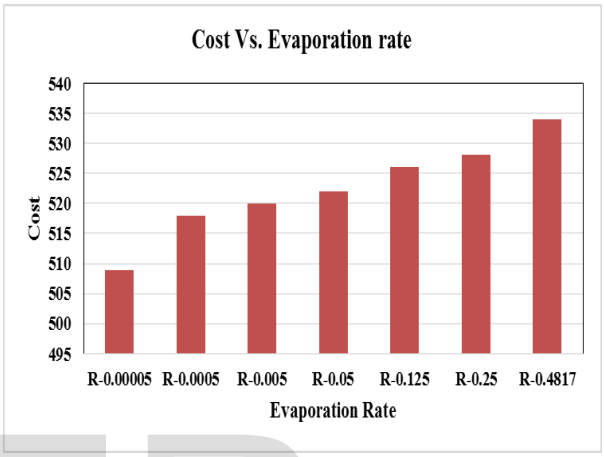


**Figure (7): Experiment 1.d results**

## V. Conclusions

ACO algorithms inspired many applications to be solved in real life problems, such as the TSP. The first example inspired by an algorithm of Ant System, was proposed solution for the well-known TSP. In this work implementation and simulation for the behavior of ACO was provided, and the best gain in performance to solve the TSP problem was given. Multiple experiments were executed to provide the best path (minimum distance) in such ways. By concentrating on main parameters which affecting the algorithm performance. Different implementation for the algorithm using main parameters, which could be tuned to gain the best performance, such as number of ants, number of cities, number of iteration and the evaporation rate. Each parameter affected the control of the execution of the algorithm in different ways, and played a significant role in the performance measures. With encouraging initial results for TSP solved by ACO algorithm in the literature, and despite the fact that ACO didn't provide the best solution for such problems usually, it still play an important role of further research both on algorithmic and application disciplines. This leads to obtain much better computational performance. The ACO metaheuristic still providing a common solution framework for common applications and state of arts problems.

## VI. REFERENCES

[1] Neto, R. F. T., Godinho Filho, M., & Da Silva, F. M. (2015). An ant colony optimization approach for the parallel machine scheduling problem with outsourcing allowed. Journal of intelligent manufacturing, 26(3), 527-538.

[2] Neto, R. T., & Godinho Filho, M. (2013). Literature review regarding Ant Colony Optimization applied to scheduling problems: Guidelines for implementation and directions for future research. Engineering Applications of Artificial Intelligence, 26(1), 150-161.

[3] Reed, M., Yiannakou, A., & Evering, R. (2014). An ant colony algorithm for the multi-compartment vehicle routing problem. Applied Soft Computing, 15, 169-176.

[4] Delévacq, A., Delisle, P., Gravel, M., & Krajecki, M. (2013). Parallel ant colony optimization on graphics processing units. Journal of Parallel and Distributed Computing, 73(1), 52-61.

[5] Cecilia, J. M., García, J. M., Nisbet, A., Amos, M., & Ujaldón, M. (2013). Enhancing data parallelism for ant colony optimization on GPUs. Journal of Parallel and Distributed Computing, 73(1), 42-51.

[6] Pedemonte, M., Nesmachnow, S., & Cancela, H. (2011). A survey on parallel ant colony optimization. Applied Soft Computing, 11(8), 5181-5197.

[7] Dorigo, M., & Stützle, T. (2010). Ant colony optimization: overview and recent advances. In Handbook of metaheuristics (pp. 227-263). Springer US.

[8] Sleit, A., Al-Akhras, M., Juma, I., & Alian, M. (2009). Applying ordinal association rules for cleansing data with missing values. Journal of American Science, 5(3), 52-62.

[9] Dorigo, M., & Stutzle, T. (2003). The ant colony optimization metaheuristic: Algorithms, applications, and advances. International series in operations research and management science, 251-286.

[10] Almobaideen, W., Al-Khateeb, D., Sleit, A., Qatawneh, M., Qadadeh, K., Al-Khdour, R., & Hafeeza, H. A. (2013). Improved stability based partially disjoint AOMDV. International Journal of Communications, Network and System Sciences, 6(05), 244.

[11] Dorigo, M., Birattari, M., & Stutzle, T. (2006). T.: Ant colony optimization-artificial ants as a computational intelligence technique. Universit Libre de Bruxelles. IRIDIA Technical report Series, Belgium.

[12] Hammo, B., Sleit, A., & El-Haj, M. (2008). Enhancing retrieval effectiveness of diacritisized Arabic passages using stemmer and thesaurus. In The 19th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS2008).

[13] Gendreau, M., & Potvin, J. Y. (2010). Handbook of metaheuristics (Vol. 2). New York: Springer.

[14] Sleit, A., Saadeh, H., Al-Dhamari, I., & Tareef, A. (2010, January). An enhanced sub image matching algorithm for binary images. In American conference on Applied mathematics (pp. 565-569).

[15]https://www.mathworks.com/matlabcentral/fileexchange/51113-ant-colony-optimization--aco--to-solve-traveling-salesman-problem--tsp-?s_tid=srchtitle

[16] Sleit, A., AlMobaideen, W., Baarah, A. H., & Abusitta, A. H. (2007). An efficient pattern matching algorithm. Journal of Applied Sciences, 7(18), 269-2695.