# U-Turning Ant Colony Algorithm powered by Great Deluge Algorithm for the solution of TSP Problem

**Saman Mohammed Almufti**

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
February 2015
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Serhan Çiftçioğlu
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering

Asst. Prof. Dr. Ahmet Ünveren
Supervisor

Examining Committee

1. Asst. Prof. Dr. Adnan Acan

2. Asst. Prof. Dr. Mehmet Bodur

3. Asst. Prof. Dr. Ahmet Ünveren

# ABSTRACT

In latest years, Optimization Algorithms have been one of the most interesting applications that can be used in order to solve tough real life problems. Real life problems could be either single or multi objective. In general Optimization techniques try to minimize an objective function for any real life problem. One of the most interesting real life problems is Traveling Salesman Problem (TSP) that is NP-hard which cannot be solved straightforwardly. Swarm Intelligence that is a field of Artificial Intelligence, uses the behaviors of real swarms to solve Optimization problems. Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Artificial Bee Colony Algorithm (ABC) are the well-known Swarm Intelligence algorithms that are generally used in solving NP-hard problems.

In this thesis, TSP problems are solved by using ACO algorithm which uses the behavior of real ants. For the betterment of the solutions found by ACO, a local search, Greate Deluge Algorithm (GDA) is used, also a new type of Ant defined as U-Turning Ant (UAnt) which returns back without completing its route guides the remainders in finding the shortest route. In this thesis it is shown that by hybridizing ACO with local search and using U-Turning Ants in ACO (U-TACO), the solutions of the given TSP problems will be improved.

**Keywords:** Traveling Salesman Problem (TSP), Ant Colony Optimization (ACO), Great Deluge Algorithm (GDA), and U-Turning Ant Colony Algorithm (U-TACO).

# ÖZ

Son yıllarda, En iyileme Algoritmaları gerçek hayat problemlerini çözmek için kullanılabilecek en ilginç uygulamalarından biri olmuştur. Gerçek hayat problemleri tek veya çok amaçlı olabilmektedir. Optimizasyon algoritmaları genellikle verilen gerçek hayat problemlerinin amaç fonksiyonlarını en aza indirmeye çalışmaktadır. En ilginç gerçek hayat problemlerinden biri polinomsal zamanda kolayca çözülemeyen Gezgin Satıcı Problemidir (GSP). Yapay Zekanın bir parçası olan Sürü Zekası gerçek hayat sürü davranışlarını kullanarak Optimizasyon problemlerine çözüm üretir. En iyi bilinen Sürü Zeka algoritmalarına örnek olarak, Karınca Koloni Optimizasyonu (KKO), Parçacık Sürü Optimizasyonu (PSO), ve Yapay Arı Koloni algoritması (YAK) gösterilebilir.

Bu tezde, GSP problemleri gerçek karınca davranışlarını kullanan KKO algoritması ile çözülmüşlerdir. Elde edilen çözümlerin iyileştirilmesi için de yerel arama algoritması olan Büyük Tufan Algoritması (BTA) kullanıldı. Ayrıca kendi rotasını tamamlamadan geri dönen yeni bir tür karınca olan U-Dönüşü yapan karıncalar (UKarınca), geriye kalan karıncaların rotalarını enkısa yoldan tamamlamalarına yardımcı olmak amacıyle, tanımlanmıştır. Bu tezde U-Dönüşü yapan karıncaları kullanan KKO algoritmsı ile birleştirilen BTA algoritmasının GSP problemlerinin çözümlerini iyleştirdiği gösterilmiştir.

**Anahtar Kelimeler:** Gezgin Satıcı Problemi (GSP), Karınca Koloni Optimizasyonu (KKO), Büyük Tufan Algoritması (BTA), ve U-Dönüş yapan Karınca Koloni Optimizasyonu (U-TACO).

## DEDICATION

*To my beloved family*

*&*

*My best friends*

# ACKNOLEDGMENT

I gratefully acknoLedge the contributions of my Supervisor Asst. Prof. Dr. Ahmet Unveren and all faculty members of Computer Engineering at Eastern Mediterranean University.

I express my warm thanks to my best friends Kamiran S. Othman and Awaz A. Shaban who always urged me to continue my study to get MSc degree.

In addition, I would like to thank the members of my family who assisted me with this project. I appreciate all their love, encouragement and support avoid me being dyslexic and catch all my goals.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| TSP | Traveling Salesman Problem |
| ACO | Ant Colony Optimization |
| GDA | Great Deluge Algorithm |
| ACO&GDA | Ant Colony Optimization Powered by Great Deluge Algorithm and 2-Opt |
| U-TACO | U-Turning Ant Colony Optimization |
| CO | Combinatorial Optimization |
| NP-hard | Non-deterministic Polynomial-time hard |
| SI | Swarm Intelligence |
| L | Level |
| $\eta_{ij}$ | Heuristic value |
| $\tau_{ij}$ | Pheromone trail matrix |
| $\alpha$ | Parameter to regulate the influence of $\tau_{ij}$ |
| $\beta$ | Parameter to regulate the influence of $\eta_{ij}$ |

# Chapter 1

# INTRODUCTION

Traveling Salesman Problem (TSP) is one of the NP-hard problems in which a Salesman wants to have a minimum distance (shorter) tour between a given set of cities and return to (starting city) his hometown (Laporte, 1992), at the end of TSP the Salesman must find the cheapest Hamiltonian cycle (shortest tour) in the given set of cities. Hamiltonian cycle is a tour that each city or node in a graph visited exactly once (Patel & Doshi, 2011). TSP is an important problem in developing and testing many new optimization techniques, generally TSP problem divided into two category Symmetric Traveling Salesman Problem (STSP) and Asymmetric Traveling Salesman Problem (ATSP) (Goyal, 2010).

Due to the difficulty of finding the optimal tour for TSP, which is one of the most known NP-hard problems, most of the optimization algorithms use a local search algorithm to find the best tour for the salesman.

In this thesis the Ant Colony Optimization Algorithm (ACO) powered by the Great Deluge Algorithm (GDA) will be used to find the optimal solution for STSP problems.

Ant Colony Optimization Algorithm uses the real ants behaviors to obtain a solution for TSP. Dorigo and his colleagues were the first who applied the idea of Ant Colony Optimization Algorithm to Traveling Salesman Problem (TSP) (Colorni, Dorigo,

Maniezzo, 1991, 1996). In Ant colony algorithms, artificial ants make their choice to add cities to the tour by measuring the amount of pheromone trail in the edge connecting cities. Great Deluge Algorithm (GDA) is adapted to TSP problem and hybridized with Ant Colony Optimization Algorithm (ACO) to improve the efficiency of ACO algorithm.

Great Deluge Algorithm (GDA) is a local search method for solving optimization problems, and it was introduced by Dueck (Dueck, 1993). GDA, like simulated annealing (SA) (Bykov, 2003) and other local search techniques, depends on iteratively replacement of a current solution by a neighborhood solution, until a stopping condition is satisfied. In this thesis 2-opt algorithm is used to find neighborhood solution which increase performance of Great Deluge Algorithm.

U-Turning Ant Colony Optimization (U-TACO) strategy is used for further improving the ACO algorithm to converge best solutions. U-TACO makes random partial tour of a specific length for the ants, and initializes the pheromone trail value of the visited cities in the partial tour. This pheromone initialization used by ACO in construction of the whole tour, at each ACOs iteration U-TACO's agent update a specific length of the best tour.

In the beginning of this chapter we gave a general introduction to the principles will be used in this thesis, the rest of the chapter contains a general description of NP-hard problems, TSP and the methods used to solve TSP problem including ACO, GDA, and 2-Opt. Other chapters of this thesis are organized as follows: in Chapter 2 we propose the methods used for the solution of TSP which illustrate the mechanism of using 2-Opt inside GDA, ACO used with GDA and 2-Opt (ACO&GDA ), and

U-Turning Ant Colony Optimization (U-TACO). Chapter 3 presents Experimental results of all proposed methods and a comparison between them. Finally Conclusion is given in Chapter 4.

## 1.1 Combinatorial Optimization

Combinatorial Optimization (CO) problems involve computing a good solution for discrete variables such as finding an optimal solution for a problem with respect to a given objective function (Schrijver, 2003). In theoretical computer science fields the Combinatorial optimization is either a minimization or a maximization of a problem (Iwr.uni-heidelberg.de, 2014), it arises in many optimization problems such as finding shortest path, planning, scheduling, time-tabling, finding models of propositional formulae, internet data packet routing, finding a minimum cost plan for a customers and many other important real-world problems (Schrijver, 2005).

## 1.2 NP-hard Problem

Non-deterministic Polynomial-time hard (NP-hard) problems are those problems which are strongly believed that their optimal solution cannot be found within a polynomial bounded computation time (Hochbaum, 1997).

Generally there are two types of methods for solving real-life optimization problems either *complete approach* or *approximate approach*. Complete approach requires exponential computing time for solving NP-hard optimization problems, whereas approximate approach which is divided in to two search approaches, *single-based search* and *population-based search*, focuses on obtaining a good solutions in relatively less time instead of finding optimal solutions which are hard to compute (Blum, 2005).

For most NP-hard problems using exact algorithms are not preferable, because they takes unbounded time, that is why most of the research often use approximate methods, which obtain a near optimal solution for NP-hard problems in a significantly short bounded time, and which are broadly known as *heuristic methods* (Sahni & Gonzalez, 1976).

## 1.3 Travelling Salesman Problem

Traveling salesman problem (TSP) is one of the widely studied problems in Mathematics and Computer Science fields. Due to its difficulty many mathematical scientists studied it, going back to 1920, when the mathematician Karl Menger first issued TSP with his colleagues in Vienna (LaLer, Lenstra, Rinnooy Kan & Shmoys, 1985), at that time TSP problem was called "Messenger Problem" by Karl Menger (Theorsociety.com, 2014). Later in 1930, mathematical community of Princeton sighted to the problem (AppleGate, Bixby, Chvatal & Cook, 1998). In 1940 the scientist mathematician Merrill Meeks Flood discusses TSP through random selection of locations in the Euclidean plane (Mahalanobis, 1940). In 1948 Flood publicized the traveling salesman problem in RAND Corporation, in 1949 the first reference containing the term “traveling salesman problem (TSP)” was reported by Julia Robinson under the name “On the Hamiltonian game (a traveling salesman problem)” (AppleGate et al., 1998). In 1954 Ray Fulkerson, Selmer Johnson, and George Dantzig descript a method for solving the TSP, they have found an optimal tour to a TSP problem of 49 cities (Wikipedia, 2014). In 1956 some heuristic methods was described by Flood including the 2-opt and nearest-neighbour algorithm which are used for finding a good tour for TSP (Theorsociety.com, 2014). In 1972 Richard M. Karp showed that TSP is one of NP-hard problems, in 1977 the optimal tour of West Germany which composed of 120 cities were found by

Groetschel (Theorsociety.com, 2014). The progress continued during the 1980s, when the exact solutions of the TSP problems up to 2392 cities were found by scientists Grötschel, Padberg, Rinaldi and others (Wikipedia, 2014). In 1991, the researcher of the University of Augsburg Gerhard Reinelt published a collection of 84 varying difficulty TSP problems under the name TSPLIB. In 2006, Cook and other researchers found an optimal tour for 85,900 city TSP problem (Wikipedia, 2014).

Travelling Salesman Problem (TSP) is an NP-hard problem in combinatorial optimization (Laporte, 1992; Johnson & McGeoch, 2002), which has a huge search space that it cannot be solve easily (Garey & Johnson, 1979; Louis & Gong, 2000). Given a set of cities $\{C_1, C_2, C_3, ...., C_n\}$ in which every city must be visited once only and return to the starting city for completing a tour such that the length of the tour is the shortest among all possible tours. For each pair of cities $\{C_i, C_j\}$ the distance of arc connecting those cities denoted by $d(C_i, C_j)$, the best tour for a salesman specified by an order permutation $(\pi)$ of cities that have minimum tour length.

Generally there are two different kinds of TSP problems, Symmetric TSP (STSP) and Asymmetric TSP (ATSP). For the STSP the distance $d(C_i,C_j) = d(C_j, C_i)$, for $n$ cities the number of possible tours is $(n-1)!/2$, whereas for ATSP, where the distance $d(C_i,C_j) \neq d(C_j, C_i)$, for $n$ cities the number of possible tours is $(n-1)!$, for large number $(n)$ of cities it is very difficult to find the exact best tours in both ATSP $(n-1)!$ and STSP $(n-1)!/2$, that is why TSP considers one of the NP-hard problems (Johnson & Papadimitriou, 1985).

Formally, the TSP is a complete weighted graph *G (N, A)* where N is the set of cities which must be visited, and *A (i, j)* is the set of arcs connecting the city (*i)* and city (*j)*. The length between city $A_i$ and $A_j$ can be represented as $d_{ij}$ (Dorigo, 2004). Thus the tour length to the given TSP problems can be found by finding the summation of the length between the cities of a permutation list as shown in formula (1.1) (Denis, 2004-2005).

$$Tourlength = \left(\sum_{i=1}^{n-1} d_{\pi(i)\,\pi(i+1)}\right) + d_{\pi(n)\,\pi(1)} \qquad (1.1)$$

Where $\pi$ is the permutation list of cities.

## 1.4 Swarm Intelligence

In computer sciences, Swarm Intelligence (SI) is the field of studying and designing efficient computational methods to solve problems using the behavior of real swarms such as birds, fish, and ants (Bonabeau et al., 1999; Kennedy et al, 2001). SI is a part of Artificial Intelligence introduced in the global optimization framework in 1989 by Jing Wang and Gerardo Beni as a collection of algorithms for controlling robotic swarm (Beni & Wang, 1989).

Swarm Intelligence issued a number of homogenous agents which interacts with each other either directly or indirectly, they communicate directly with each other by using audio or visual tools, as the honey bees communicate by waggle dance; indirect communication refers to as stigmergy (Dorigo, Bonabeau, & Theraulaz, 2000). Grasse first introduced the concepts stigmergy to the processes when an insect makes change in the environment around it, and the other insects respond to that change and adapt themselves to the new environment, such as in ant colonies when an ant deposit pheromone in its way to the food, leads other ants to follow that way

(Grasse, 1959). Examples of swarm intelligence methods are Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Particle Swarm Optimization (PSO), Bacterial Foraging, Artificial Immune System, Stochastic diffusion search, Cat Swarm Optimization, Gravitational search algorithm, Bat algorithm, and Glowworm Swarm Optimization (Merkle & Middendorf, 2008).

## 1.5 Ant Colony Optimization (ACO)

Ant Colony Optimization algorithms (ACO) belongs to swarm intelligence (SI) field methods. Ant System (AS) was first represented in 1992 by Marco Dorigo in his PhD thesis as a nature-inspired metaheuristics for solving hard combinatorial optimization (CO) problems (Dorigo, 1992). Later in 1997 developed algorithms of Ant Colony System (ACS) were published by Dorigo and Gambardella (Dorigo & Gambardella, 1997). The progress continued when one year later, in 1998, Dorigo introduced Ant Colony Optimization algorithms (ACO) for the first time in a conference (Dorigo, 1998; Dorigo, Maniezzo, & Colomi, 1991; Dorigo, 1992; Dorigo & Di Caro, 1999; Dorigo & Stützle 2004).

Over years, many algorithm designed that depend on Ant behaviors, such us Ant System (AS), Ant colony Optimization (ACO), Ant-Q, and Ant Colony System (ACS) (Dorigo, Caro and Gambardella, 1999). Generally the algorithms that depend on ant behaviors and specifically Ant colony optimization, over years ago has been adapted to solve many optimization problems in real life such as Mobile and Wireless Sensor Networks (WSNs), vehicle routing problem, Network Security, Computer Architecture, Data Mining and many other problems (Geetha & Srikanth, 2012).

### 1.5.1 Ant Colony Optimization (ACO) Metaheuristic

Ant Colony Optimization (ACO) is a metaheuristic algorithm for solving combinatorial optimization (CO) problem (Dorigo & Di Caro, 1999).

Metaheuristic algorithms uses some basic heuristics in the nature to escape from local optimal to build an optimal solution for a real life problem (Denis, 2004-2005), some metaheuristic algorithms start from an empty solution then add nodes (elements) to build a solution, which is called *constructive heuristics*, when other algorithms start from a weak complete solution and modify it iteratively to obtain a better solution, it is called *local search heuristics* (Blum & Roli, 2003).

ACO is a *constructive heuristic* algorithm, its metaheuristics is an inspiration of real ant's behaviors in the nature, which use the intelligence and strong attraction of swarms in finding food (Deneubourg, Aron, Goss, & Pasteels, 1990). It basically depends on pheromone trail updating and the following of other ants to the pheromone smell: ants deposit pheromones in their way to the food source which takes attention of other swarm ants to the food source, and the way that they should take to the food. Artificial ants update pheromone forwarding to the food, the increasing of pheromone in one path and the pheromone evaporation process which decrease pheromone intensity in other paths avoids unlimited trails accumulation over some components (Mute-net.sourceforge.net, 2015).

### 1.5.2 Real Ant Behavior and Capabilities

In the nature real ants which are '*almost blind'* insects are capable to figure out the shortest path connecting their nest with a source of food Figure 1. Ants communicate to each other without using any visual indication (Hölldobler & Wilson, 1990) they indirectly communicate by pheromone trails, which is an odorous chemical substance

that ants may deposit and smell (Goss, Aron, Deneubourg & Pasteels, 1989). The term "pheromone" was first introduced in 1959 by P. Karlson and M. Lüscher, that comes from Greek word “pherein” which means “to transport” and “hormone” which means “to stimulate” (Karlson & Lüscher, 1959).

Ants lay pheromone in the way that takes them to the food, which make other ants follow that path. The amounts of pheromone on a single path depends on the length between food source and ant’s nest, in which the way having greatest pheromone density is the shortest way to the food, pheromone directly increases with the number of ants following that path (Mute-net.sourceforge.net, 2015).

The behavior of real ants searching for food in nature Figure 1 (Mute-net.sourceforge.net, 2015):

- First, each ant randomly lays down a pheromone trail in their path searching for food source.
- If any of the ants finds a food, it returns to the nest laying down a pheromone trail
- If in any path pheromone is increased, the other ants follow that path.

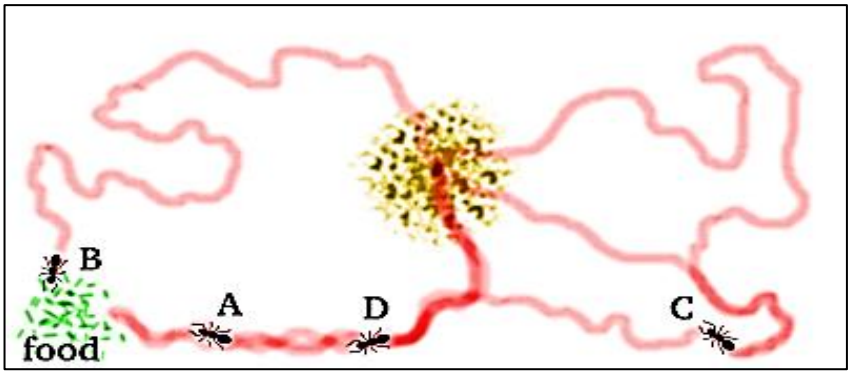

Figure 1: Behavior of real ants (Mute-net.sourceforge.net, 2015)

Ants can find a new short path in the case of absence of the old one after inserting an obstacle (Beckers, Deneubourg & Goss, 1992). Consider Figure (2A) a straight way of ants from nest to food source, in Figure (2B) an obstacle inserted in their way, here ants arrive at a decision point and must make a choice whether to turn left or right, in Figure (2C) half of ants takes one side of the obstacle and other ants takes the other side of the obstacle, all ants move with approximately same speed and approximately the same amount of pheromone deposit which lead to accumulate a greater amount of pheromone trail in the shorter path per unit time, as the results showen in Figure (2D). Ants make their decision according to the amount of pheromone and choose a shortest way between food source and the nest (Dorigo, Maniezzo, & Colomi, 1991).

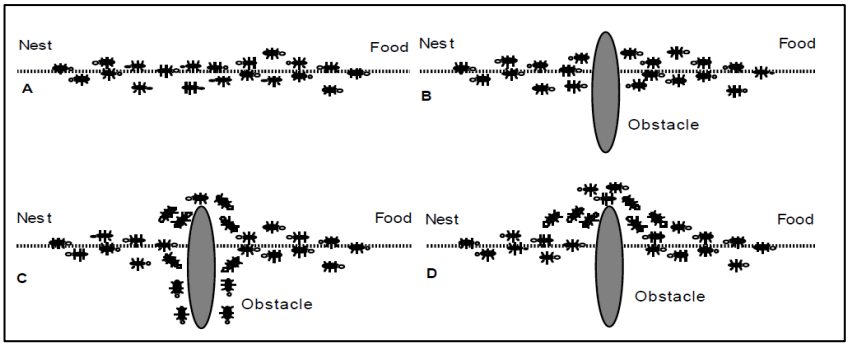

Figure 2: Ants are capable of finding new path (Dorigo, Maniezzo, & Colomi, 1991)

### 1.5.3 ACO and TSP

The traveling salesman problem (TSP) has an important role in Ant System (AS) algorithms in general. The first appearance of Ant System was tested on the travelling salesman problem in 1992 by Marco Dorigo (Dorigo, 1992).

The Ant Colony Optimization Algorithm agents (Artificial Ants) move from one city to another city until completing their tours and then return back to the starting city on

TSP graph, they iteratively construct solutions and deposit pheromone on the arc connecting cities one by one.

ACO algorithm follows many steps iteratively to construct best solutions for TSP problems, (i) each ant randomly chooses a city at the beginning, (ii) For each ant: ant chooses the next city according to the heuristic information and pheromone value on the edge between current city and the next one, and then pheromone update on the edges used, (iii) After all ants construct their path (solution) a general pheromone trail updates, (iv) A path, which has the best pheromone amount is the best solution for the TSP problem (Dorigo & Gambardella, 1997). Figure 3, shows the steps of ACO algorithm that solves TSP problem.

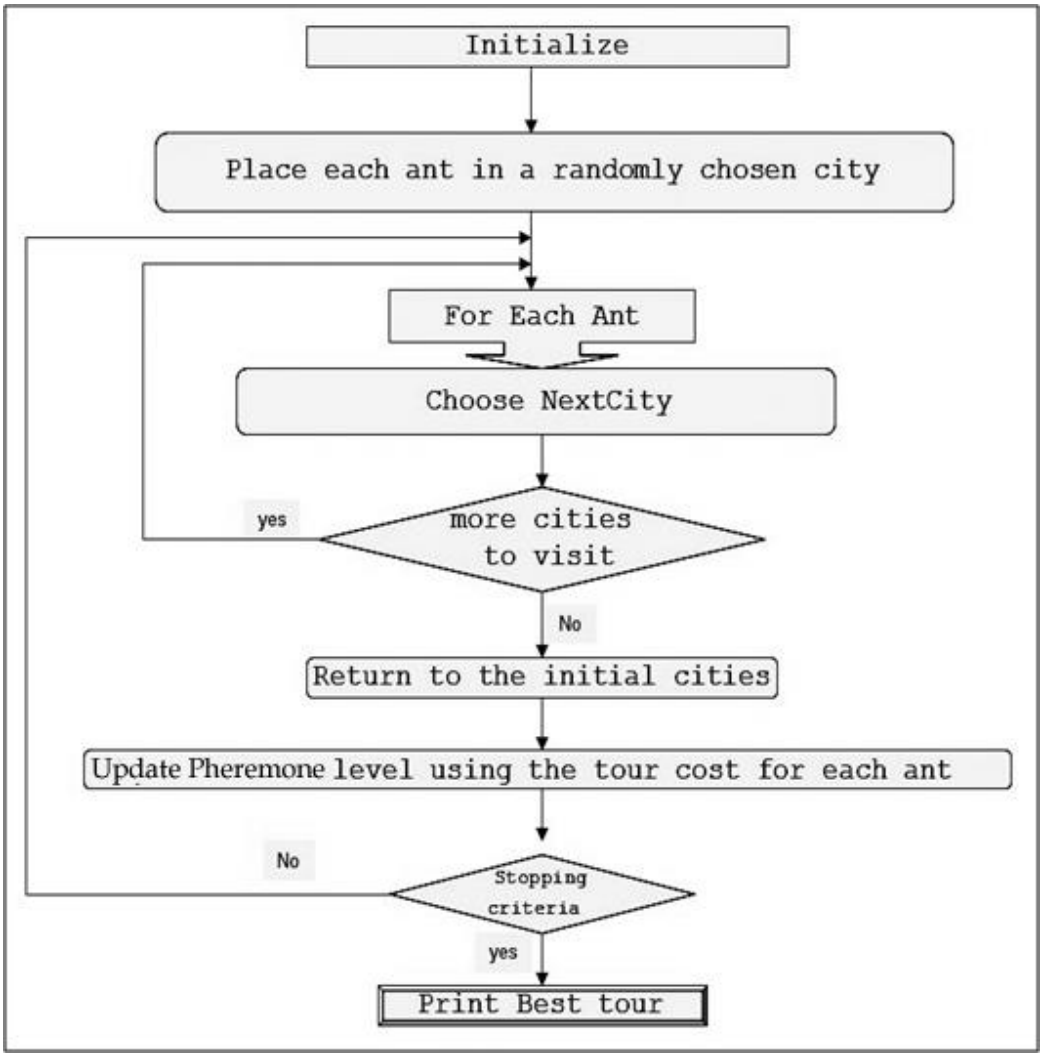

Figure 3: Flowchart of ACO algorithm for TSP (Samaiya & Samaiya, 2012)

Practically ACO algorithm used for TSP problems agents (Artificial ants) are equal to the number of cities of a TSP problem ($M == N$, $M$ number of ants and $N$ number of cities). Each ant makes a solution tour, initially all ants locate in cities randomly, then every ant chooses the next city upon a probability based function depending on both pheromone trail accumulated on edge and heuristic value. The formula used for choosing next city called *random proportional rule* (1.2), shows the probability of ant *(k)* that locate in the city *(i)* to visit city *(j)* (Denis, 2004-2005).

$$P_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{i \in \mathcal{N}_i^k}[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}} \quad \text{, if j} \in \mathcal{N}_i^k \tag{1.2}$$

Where $\alpha$ and $\beta$ are variables that determine the influence of pheromone trail and heuristic information, where $\alpha$ is a parameter to regulate the influence of pheromone trail ($\tau_{ij}$) and $\beta$ is a parameter to regulate the influence of heuristic value ($\eta_{ij}$), $\alpha$ and $\beta$ effects in the process of choosing next city as shown below.

- $\alpha < \beta$ The closest city is more likely to be chosen.
- $\alpha > \beta$ The arcs that have more pheromone intensity are more likely to be used.
- $\alpha = 0$, heuristic value is used for the choosing without pheromone.
- $\beta = 0$, pheromone is used for the choosing without heuristic value, which may cause poor result.

$\eta_{ij} = 1/d_{ij}$, is a priori available heuristic value, ($d_{ij}$) is the distance between cities (*i* and *j*), ($\tau_{ij}$) is the pheromone trail matrix, and ( $\mathcal{N}_i^k$) is the set of the neighborhood that ant (*k)* has not visited yet (Denis, 2004-2005).

The probability that ant *(k)* choose *arc (i, j)* increases with the value of pheromone trail ($\tau_{ij}$) and heuristic value ($\eta_{ij}$).

In this thesis ACO uses $\alpha = 1 \; and \; \beta = 2$ in the case of 20 ACO iterations, and it uses $\alpha = 1 \; and \; \beta = 5$ in the case of 100 ACO iterations, the initial value of pheromone matrix usually set to a small value which is greater than zero (1.3).

$$\forall \, (i,j) \Leftrightarrow \; \tau_{ij} = \tau_0 \, , where \; \tau_0 > 0 \tag{1.3}$$

Ants choose the city, which contains larger amounts of pheromone on the connecting edges between the current city and the next one.

Each ant after adding new city to its tour makes a local pheromone update (1.4), by adding the $\Delta\tau_{ij}$ to the old $\tau_{ij}$

$$\tau_{ij}new=\tau_{ij}old+\Delta\tau_{ij} \tag{1.4}$$

Where the value of $\Delta\tau_{ij}$ can be found as shown in (1.5), $\Delta\tau_{ij}$ changes according to the length ($L_k$) tour (1.6)

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k} \tag{1.5}$$

$$\Delta\tau_{ij}^{k}=\begin{cases}\frac{Q}{L_k}, \text{ if ant k uses arc } (ij)\\ 0 \quad \text{otherwise}\end{cases} \tag{1.6}$$

Where $\tau_{ij}$ is the pheromone trail on the *arc (ij)*, m is number of ants, and $\Delta\tau_{ij}$ is the summation of change in the $\Delta\tau_{ij}^{k}$, which is equal to a constant value ($Q$) over current length of the tour constructed by ant *(k)*, in this thesis we use value ($Q=1$).

The process of pheromone trail update continues until all ants complete their tour by visiting all cities and returning to the start city, after all ants terminate the construction of tour it is followed by pheromone evaporation (1.7) and a global pheromone update (1.4) for the shortest tour. The shortest tour will have the greatest amount of pheromone (Denis, 2004-2005).

Pheromone evaporation reduces the amount of pheromone in arcs which leads to gradually disappears of pheromone trail in the unused arc which makes the ants to follow one path.

$$\tau_{ij}new = (1 - \psi).\ \tau_{ij}old \quad (1.7)$$

Whereas $(0 < \psi < 1)$ is a constant quantity used for reducing pheromone trail, here we used $(\psi = 0.5)$ that decrease $\tau_{ij}$ value by 0.5 which iteratively lead to disappear pheromone trail in unused arc. Figure 7, shows the ACO Pseudo-code.

1. {Initialization}

    Initialize $\alpha, \beta, and\ \psi$

    Initialize $\eta_{ij}$

    Initialize $\tau_{ij}$ (1.3)

2. {Construction}

    For each ant *k* (currently in state $_i$) do

        Chose next city (state $_j$) (1.2) and append it to the tour

3. {Pheromone Trail update}

    {

        Calculate $\Delta\tau_{ij}^{k}$ (1.6) for calculating $\Delta\tau_{ij}$ (1.5)

        Update pheromone $\tau_{ij}$ (1.4)

    }

4. {Pheromone evaporation}

    Decreasing pheromone intensity by $\psi$ value (1.7)

5. {End}

Figure 4: ACO Pseudo-code (Denis, 2004-2005).

## 1.6 Local Search Methods

Local search metaheuristics are the most successful approaches and the most used methods for solving combinatorial optimization problems over the last few years.

Local search refers to a group of methods which depend on a neighborhood (Johnson, 1990), which are a set of candidate states, that are connected directly to the current state and can be reached by a single move for finding a best solution for computationally hard optimization problems. Its methods are iteratively improve the ordering of current solution states by performing simple modifications on the current solutions to obtain a new solution, the improvement continues until some stopping conditions has been satisfied or when there is no better solution in the given neighborhood (Russell & Norvig, 2003).

The most famous and widely used Local search algorithms is *k-opt* which takes an initial tour and improves it by making flips in the tour to obtain a better tour (Chandra, Karloff, & Tovey, 1994), it is the basics of *2-opt* (Croes, 1958), *3-opt* (Lin, 1965) and *Lin-Kernighan (LK)* (Lin, & Kernighan, 1973). A more complicated local search method that could have other Local search method imbedded inside it, is called Great Deluge Algorithm (GDA), its procedure is finding a better solution from the neighborhood of the initial solution, and then it iteratively improving solution (Dueck, 1993). Another local search method which is based on Hill-Climbing, includes some kind of intelligent in it, and is known as Tabu Search (TS), its procedure based on saving the previous moves in a list called ‘tabu list’ which is used to avoid cycling (Glover,1986). Many other local search algorithms are used in computer field to escape from local optimality which take unbounded time, one of

the most known algorithms is Simulated annealing (SA), a local search algorithms which tries to improve a solution tour iteratively, using a temperature *(T)* factor, thus, before each iteration *T* value decreases. SA uses another random parameter (*i*), if (*i*>*T*), SA uses a heuristic in choosing the next solution node, otherwise it makes a random choice (Mitra, Romeo & Sangiovanni-Vincentelli, 1985).

### 1.6.1 Great Deluge Algorithm

Great Deluge Algorithm (GDA) is a single-based approximate approaches using neighbors strategies for solving optimization problems, was first introduced by Dueck in 1993 as an alternative enhanced method of the Simulated Annealing, similarly to the simulated annealing and hill-climbing algorithms. It depends on replacing the current best solution by a new solution from set of neighborhoods, replacing process continues until the solution values become equal or better than the *Level* (*L)* value, which bounds the feasible region of the search space (Dueck, 1993).

Great Deluge Algorithm (GDA) controls the search space using a boundary *Level* (*L)* which does not depend directly on the current best solution, at the beginning the *level (L)* value equal to the initial value of function evaluation (cost function), then in each iterations it monotonically increases or decreases (according to optimization problem: maximization or minimization) by the *ΔL*. *ΔL* value is the only input parameter for GDA. This decreasing/increasing process controlling the search processes in the Great Deluge Algorithm, which drive the algorithm to word is the best optimal solution (Bykov, 2003).

During the processing GDA accepts best solution in the case of obtaining an improvement in the function evaluation value of a neighborhood solution which is better than current best solution, or it may accept a worse solution in some cases if

the neighborhood solution is worse than the current solution but better than *level (L).* GDA algorithm has been used to solve many optimization problems with relatively high performance, such as for Examination Timetabling (Abdullah & Burke, 2006; Landa-Silva & Obit, 2009). Figure 5, Shows a Detailed description of Great Deluge Algorithm (GDA) steps.

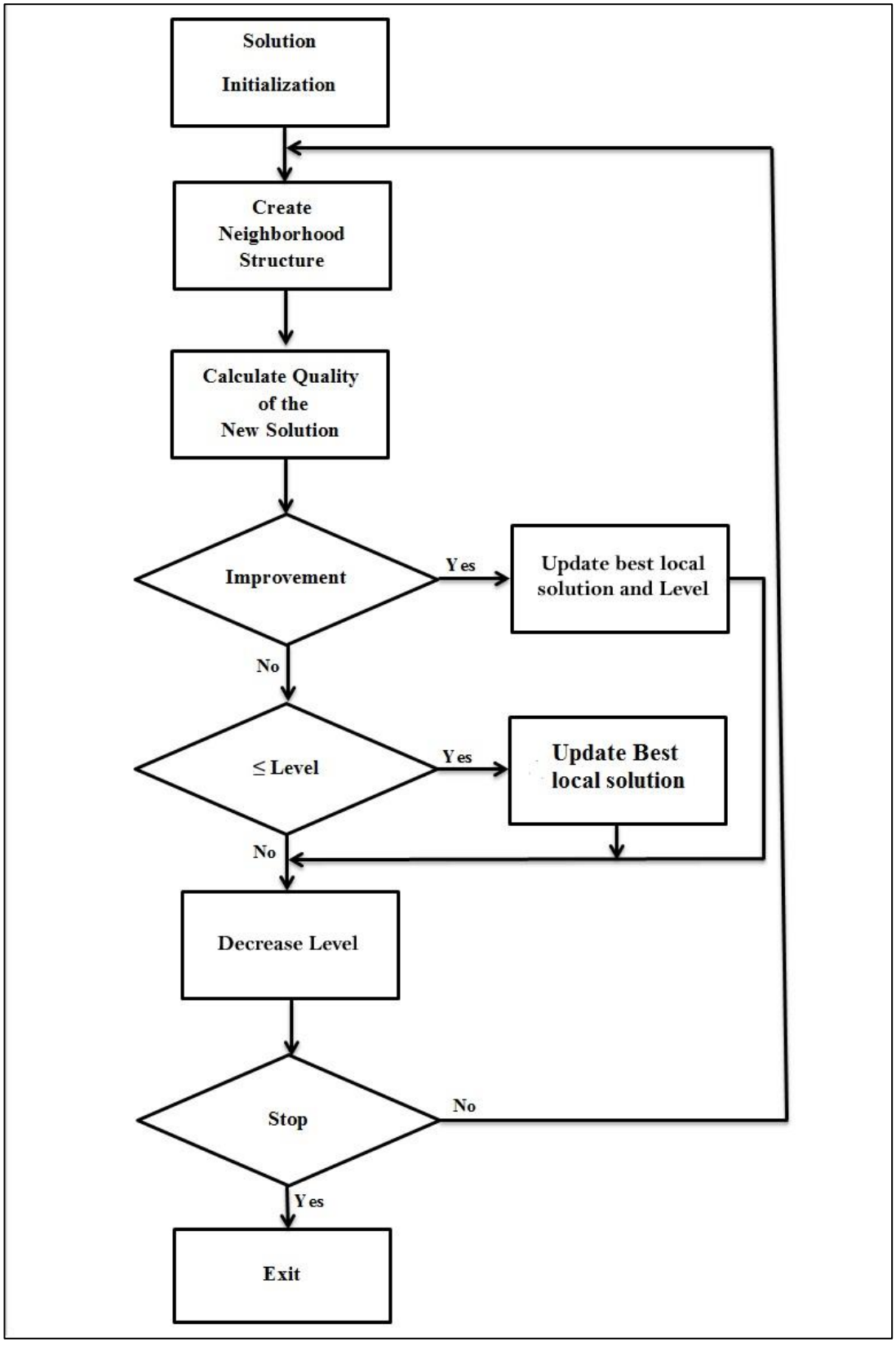

Figure 5: Flowchart of Great Deluge Algorithm (GDA) (Jaddi & Abdullah, 2013)

### 1.6.2 2-OPT Algorithm

*2-opt* is among the simplest and best known local search algorithms which was one of the first successful algorithms used to improve a solution tour of TSP (Croes, 1958). It easily takes the current tour then improves it by iteratively removing two arcs (edge) from the tour, which divides the tour into two parts, then reconnects the two parts of the tour in an opposed direction; the new tour is accepted if its fitness is less or equal to the current tour; this continues until all possible tours are being tested (Laporte, 1992; Bentley, 1992).

Figure 6, below shows an example of selected edges for deleting and reconnecting it, the initial tour on the right {1,2,3,4,5,6,7,8} after applying 2-opt algorithm by removing two blue edges in the right figure, and the reconnected tour by two red edges in the left figure obtaining a new tour{ 1,2,6,5,4,3,7,8,} (Devx.com, 2015).

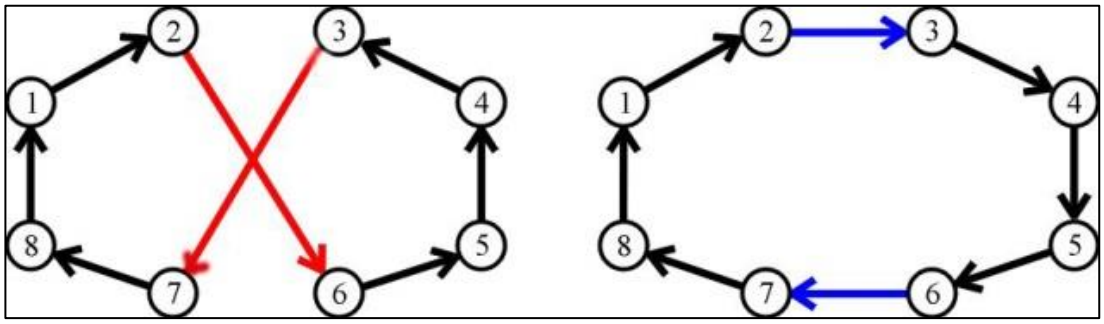


Figure 6: 2-opt example (Devx.com, 2015)

Figure 7, Shows 2-Opt algorithm steps in details.

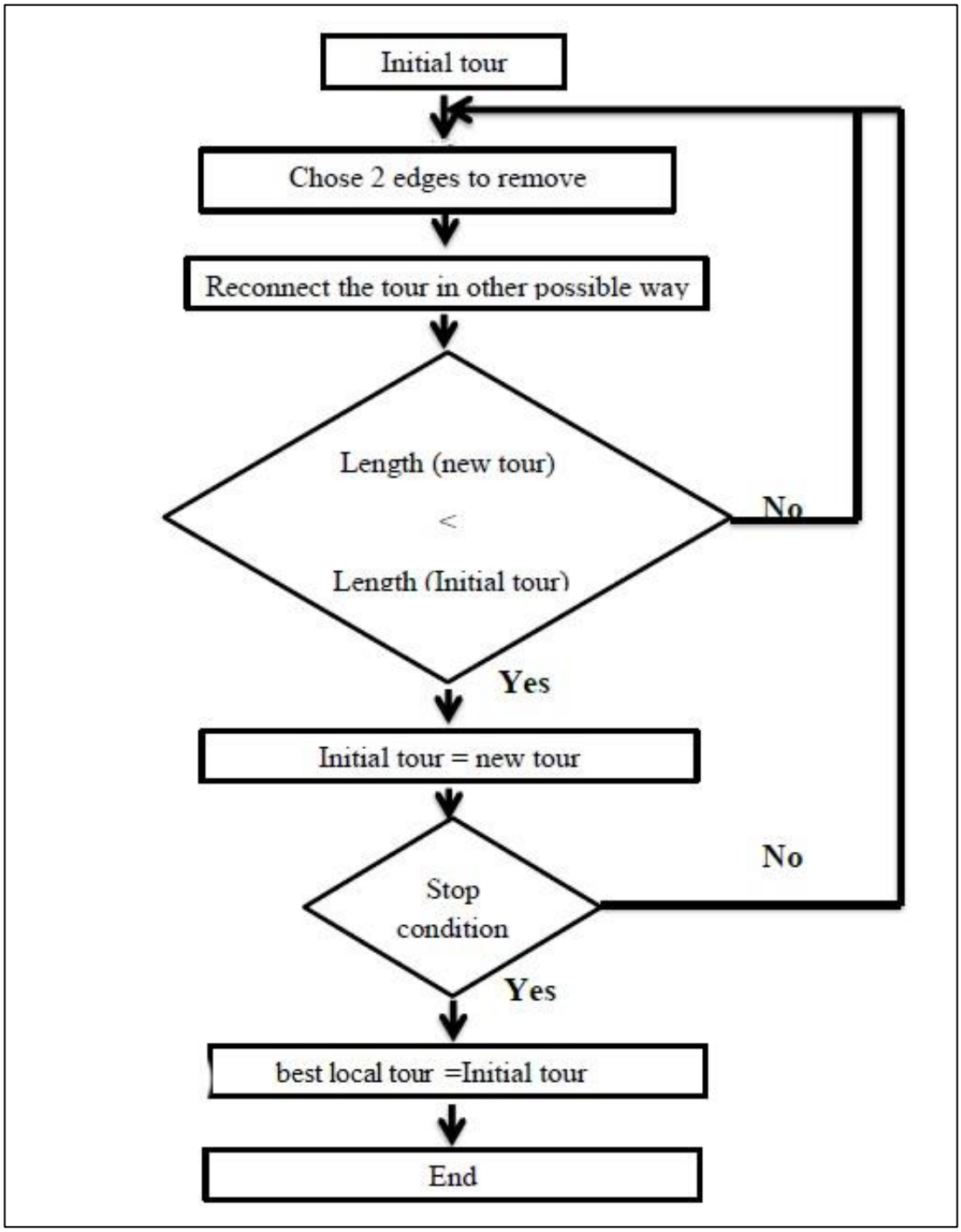


Figure 7: 2-opt flowchart (Misevičius et al., 2007)

The algorithm depends on iteratively removing long edges and applying changes to the tour, the improving process continues in such a manner that each removing and reversing operations reduces the length of the current tour, until a tour is reached for which no removing and reversing operation can proceed any improvement in the tour.

The removing and reversing operations depend on the length of edges, such as if the current tour contains two edges (first connecting city *A* with city *B* and the other connecting city *C* and *D*), the removing and reconnection of edges depend on a mathematical formula (1.8).

$$newlength = [dist(A,B) + dist(C,D)] - [dist(A,D) - dist(B,C)] \quad (1.8)$$

If the *neLength* value is negative, it means that the length of new edges (*A*⇔*D* and *B*⇔*C*) is less than the length of old edges (*A*⇔*B* and *C*⇔*D*), the new connection edges accepted and applied the changes to current best tour; otherwise, if *neLength* is positive, it means that the new tour have a length longer than the current tour, the new connection is neglected and the algorithm continues testing other edges until stopping condition is satisfied as shown in Figure 8 (Panyam, 2011).

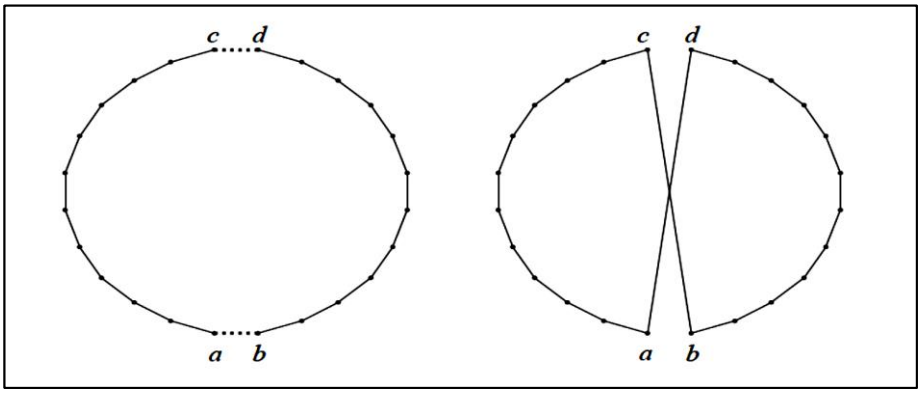


Figure 8: 2-Opt tour construction (Panyam, 2011)

Figure 9, shows the pseudo code of 2-Opt Algorithm

1. {Initialization}

    Initialize current tour

2. {Improving}

    Loop (stop condition)

    {

        Find two edges ($A \Leftrightarrow B$) and ($C \Leftrightarrow D$) for removing and reversing.

        Calculate *neLength*  (1.8)

        if *neLength* <0

            {Apply changes to current tour} removing

            $(A \Leftrightarrow B)$ and  $(C \Leftrightarrow D)$ and revers the tour ($A \Leftrightarrow D$ and $B \Leftrightarrow C$)

3. {End}

Figure 9: 2-Opt Pseudo-code (Misevičius et al., 2007)

# Chapter 2

# METHODS USED FOR THE SOLUTION OF TSP

## 2.1 Great Deluge Algorithm (GDA) and 2-Opt Algorithm

Great Deluge Algorithm (GDA) procedure finds a better solution from the neighborhood of the initial solution, then it iteratively finds a new solution by using 2-Opt local search algorithm, and compares the new solution fitness (*f(NewTour)*) with the fitness of a current best tour ( *f(BestTour)*), and the *level (L)* value. *L* is value initially equal to the initial best tour fitness (*L=f(initial BestTour)*) then it increases or decreases according to applied problem. GDA accepts new solution in two cases:

Case 1: If *f(NewTour)< f(BestTour),* the new Tour is accepted, the current best tour will be changed with the new tour (*BestTour=NewTour*), and the level (*L*) will be updated (increased for maximization /decreased for minimization) (2.1).

$$L=L\pm\Delta L \tag{2.1}$$

$$\Delta L= (f(BestTour)-Optimalrate)/(NumOfIteGDA) \tag{2.2}$$

Where $\Delta L$ can be found (2.2), it changes according to the *f(BestTour),* number of GDA iteration (*NumOfIteGD*) and Optimal rate (*Optimalrate),* which refers to as the optimal tour fitness of various problems (TSP tours).

Case 2: If *f(NewTour)< L),* the new Tour is accepted, the current best tour will be changed with the new tour (*BestTour=NewTour*) without updating level (*L*).

Figure 10, shows the pseudo code for GDA that uses 2-Opt.

```
1. {Initialization}
        Initialize initial tour
        Initialize L, Optimalrate, NumOfIteGD
        Initialize ΔL    (2.2)
2. {Improving}
        Loop (stop condition)
        {
                Finding neighborhood solution by (2-Opt algorithm)
                Calculate f(NewTour)
                        If f(NewTour)< f(BestTour)
                                {
                                        BestTour=NewTour
                                        L=L±ΔL
                                }
                        If f(NewTour)< L
                                BestTour=NewTour
        }
3. {End}
```

Figure 10: GDA Pseudo-code (AL-MILLI, 2014)

## 2.2 Ant Colony Optimization and Great Deluge Algorithm with 2-Opt (ACO & GDA)

ACO uses GDA to improve its current tour. Every ACO iterations, ACO construct a best tour by Ants, the current best tour is used by GDA as an initial tour, GDA which uses 2-Opt Algorithm inside it, further improves the tour, and then it is followed by an updating of pheromone trail on the ACO tour which has been improved by GDA.

1. {Initialization}

Initialize $\alpha, \beta, \psi, \eta_{ij}$, and $\tau_{ij}$ (1.3)

LOOP

{

2. {Construction}

For each ant k (currently in state $_i$) do

Chose next city (state $_j$) (1.2) and append it to the tour

3. {Improving}

Improving ACO current best tour by GDA (explained in (2.1))

4. {Pheromone Trail update}

Calculate $\Delta\tau_{ij}^{k}$ (1.6) for calculating $\Delta\tau_{ij}$ (1.5)

Update pheromone $\tau_{ij}$ (1.4)

5. {Pheromone evaporation}

Decreasing pheromone intensity by $\psi$ value (1.7)

6. {End}

} END LOOP

Figure 11: ACO & GDA Pseudo-code

## 2.3 U-Turning Ant Colony Optimization (U-TACO) Powered by GDA

U-Turning ACO is a simplified version of ACO algorithm which is applied to ACO as a prefix method to obtain a better solution.

Generally U-TACO has a procedure exactly the same to ACO in construction of tours, pheromone updates, and pheromone evaporations. The only difference is that in U-TACO artificial ants (UAnt) instead of visiting all cities in the TSP graph, which takes long time, make a specific number of iteration, visit a limited number of cities, and return to their starting city feedback on the same way as shown in Figure 12.

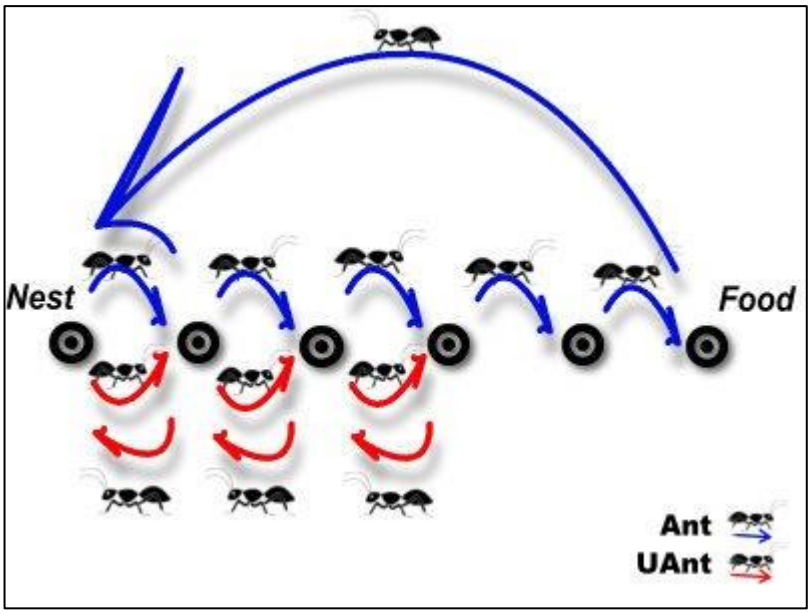


Figure 12: Ant and UAnt tour

The UAnts mission is not to find the best optimal tour, it is used just to initialize the pheromone trail matrix, which is later used by Ants to specify their required mission (finding optimal TSP tour).

At the beginning U-TACO should determine the number of iteration and the length of tour that U-Turning artificial ants (UAnt) should take, it can take any length (*1< U-TACO tour length< number of cities*). In this thesis, we have tried three different values for the U-TACO tour length *1/4 of number of cities* , *1/2 of number of cities* , and *3/4 of number of cities*.

UAnts construct their tours based on *random proportional rule* (1.2), it can have various value for ($\alpha, \beta$), it may take values different from the values used by Ant, we use values that is equal to Ant values ($\alpha=1, \beta=2$).

UAnt use the same pheromone trail update formula (1.4) that is used by Ant in ACO, here we use a value for ($Q=3$), which is used in updating formulas that are explained before in formula (1.6)

The difference in ($Q$) value leads to changes in both $\tau_{ij}new=\tau_{ij}old+\Delta\tau_{ij}$... (1.4), and $\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}$ … (1.5), this difference in the updating value makes the pheromone deposit more quickly on the best tour, and leads the other UAnt to follow it.

After UAnt make their tour of specific length, they return to their starting city by the same path (feedback), making a further updating in pheromone trail matrix, then there is a final process ending UAnts mission – pheromone evaporation (1.7).

After UAnt finish their mission (making partial tour, and initializing pheromone trail matrix ($\tau_{ij}$)) Ants mission starts to make a complete tour and finding optimal tour or best tour in ACO explained before in (1.5.3). Ants make their choice based on the pheromone matrix ($\tau_{ij}$) obtained by UAnts, then Ant make changes on the ($\tau_{ij}$)

matrix, and in every ACO iterations UAnt make further pheromone trail update to a specific length of ACO best tour.

1. {Making partial tour by UAnt}
    a. {Initialization UAnt}
        Initialize $\alpha, \beta, \psi, \eta_{ij}$, and $\tau_{ij}$ (1.3)
    Loop
    {
        b. {Construction partial tour}
        For each ant k (currently in state $_i$) do
            Chose next city (state $_j$) (1.2) and append it to the tour
        c. {Pheromone Trail update}
            With Q=3, Calculate $\Delta\tau_{ij}^{k}$ (1.6) and $\Delta\tau_{ij}$ (1.5)
            Update pheromone $\tau_{ij}$ (1.4)
        d. {Pheromone evaporation}
            Decreasing pheromone intensity by $\psi$ value (1.7)
        e. {End}
    } END LOOP
2. {Construction}
    Construct complete tour by ACO & GDA (explained in (2.2))
    UAnt update pheromone of a specific length of ACO tour
3. {End}

Figure 13: U-TACO Pseudo-code

# Chapter 3

# EXPERIMENTAL RESULTS

This section presents the experimental results obtained from Ant colony optimization (ACO), Ant colony optimization (ACO) powered by Great Deluge algorithm (ACO&GDA) and U-Turning Ant colony optimization (U-TACO) algorithms, for different symmetric traveling salesman problems (STSP) selected from (TSPLIB95).

## 3.1 Experimental Results with 20 ACO Iterations

This section shows the results of ACO, ACO&GDA, and U-TACO for different symmetric traveling salesman problems (STSP) selected from (TSPLIB95). All problems were solved with 20 ACO iterations and the parameter that determine the influence of pheromone trail $\alpha = 1$ and heuristic information $\beta = 2$, for each TSP problem number of ants equal to number of cities.

### 3.1.1 ACO Results

Table 3.1 shows the results of ACO algorithms with 20 ACO iterations. It can be seen from this table that it is not possible to find any optimal solution for the given TSP problem from (TSPLIB95).

Table 3.1: ACO results with 20 iterations

| TSP problem | Optimal Tour | ACO | Time | # of function evaluations | Optimal foundation |
|---|---|---|---|---|---|
| **att48** | 10628 | 12012 | 8.751971s | 960 | Not |
| **att532** | 27686 | 32589 | 2057.607684s | 10640 | Not |
| **berlin52** | 7542 | 8092 | 11.532953s | 1040 | Not |
| **Lin105** | 14379 | 16213 | 37.226899s | 2100 | Not |
| **Ch150** | 6528 | 6871 | 77.035097 s | 3000 | Not |
| **Eil51** | 426 | 472 | 14.233584s | 1020 | Not |
| **Eil76** | 538 | 580 | 21.822042 s | 1520 | Not |
| **Eil101** | 629 | 746 | 36.084189s | 2020 | Not |
| **fl417** | 11861 | 13443 | 1044.515205s | 8340 | Not |
| **kroA100** | 21282 | 24698 | 33.873039s | 2000 | Not |
| **kroA150** | 26524 | 30669 | 84.927355s | 3000 | Not |
| **kroA200** | 29368 | 34543 | 157.599577s | 4000 | Not |
| **kroB100** | 22141 | 25856 | 37.269276s | 2000 | Not |
| **kroB150** | 26130 | 30320 | 82.397767s | 3000 | Not |
| **kroC100** | 20749 | 23342 | 33.827723s | 2000 | Not |
| **kroD100** | 21294 | 24465 | 35.068680s | 2000 | Not |
| **Lin318** | 42029 | 48250 | 503.068268s | 6360 | Not |
| **Pr76** | 108159 | 124032 | 20.916123s | 1520 | Not |
| **Pr107** | 44303 | 46389 | 40.363483s | 2140 | Not |
| **Pr124** | 59030 | 65145 | 53.535056s | 2480 | Not |
| **Pr136** | 96772 | 111739 | 65.715502s | 2720 | Not |
| **Pr144** | 58537 | 59553 | 75.360905s | 2880 | Not |
| **Pr152** | 73682 | 78784 | 82.526424s | 3040 | Not |
| **Rat99** | 1211 | 1437 | 33.232457s | 1980 | Not |
| **Rd100** | 7910 | 9386 | 37.157033s | 2000 | Not |
| **St70** | 675 | 742 | 15.791204s | 1400 | Not |
| **Tsp225** | 3916 | 4578 | 204.455153s | 4500 | Not |

### 3.1.2 ACO&GDA Results

By using ACO&GDA algorithm, which has been run with 20 iterations on the same collection of TSP problems that is used in ACO. The Table 3.2 shows the ACO&GDA results, it can be seen that the optimal tour for four TSP problems are founded.

Table 3.2: ACO&GDA results with 20 iterations

| TSP problem | Optimal Tour | ACO&GDA | Time | # of function Evaluation | Optimal foundation |
|---|---|---|---|---|---|
| **att48** | 10628 | 10628 | 65.845424 s | 192960 | Iteration=7<br>Time=24.618300 s<br>Evaluation=67536 |
| **att532** | 27686 | 28321 | 28679.2511s | 2138640 | Not |
| **berlin52** | 7542 | 7657 | 21.190134 s | 209040 | not |
| **Lin105** | 14379 | 14434 | 448.722319s | 422100 | Not |
| **Ch150** | 6528 | 6554 | 345.672009 s | 603000 | Not |
| **Eil51** | 426 | 426 | 81.920737s | 205020 | Iteration=3<br>Time=20.193626 s<br>Evaluation=30753 |
| **Eil76** | 538 | 543 | 57.488787 s | 305520 | Not |
| **Eil101** | 629 | 630 | 112.080519s | 406020 | Not |
| **fl417** | 11861 | 12038 | 15797.0384s | 1676340 | Not |
| **kroA100** | 21282 | 21296 | 117.925550s | 402000 | not |
| **kroA150** | 26524 | 26751 | 353.030522s | 603000 | Not |
| **kroA200** | 29368 | 29591 | 818.688881s | 804000 | Not |
| **kroB100** | 22141 | 22258 | 110.074386s | 402000 | Not |
| **kroB150** | 26130 | 26292 | 344.846879s | 603000 | Not |
| **kroC100** | 20749 | 20798 | 110.332764s | 402000 | Not |
| **kroD100** | 21294 | 21395 | 112.804903s | 402000 | Not |

| | | | | | |
|---|---|---|---|---|---|
| **Lin318** | 42029 | 42670 | 7642.12961s | 1278360 | Not |
| **Pr76** | 108159 | 108159 | 60.225353s | 305520 | Iteration=14<br>Time=44.654503 s<br>Evaluation= 213864 |
| **Pr107** | 44303 | 44303 | 127.697276s | 430140 | Iteration=2<br>Time= 17.977372 s<br>Evaluation= 43014 |
| **Pr124** | 59030 | 59087 | 188.399575s | 498480 | Not |
| **Pr136** | 96772 | 102323 | 255.565197s | 546720 | Not |
| **Pr144** | 58537 | 58636 | 284.115236s | 578880 | Not |
| **Pr152** | 73682 | 73880 | 353.679060s | 611040 | Not |
| **Rat99** | 1211 | 1221 | 107.793133s | 397980 | Not |
| **Rd100** | 7910 | 7951 | 116.764002s | 402000 | Not |
| **St70** | 675 | 677 | 45.134547s | 281400 | Not |
| **Tsp225** | 3916 | 3942 | 4379.11708s | 904500 | Not |

### 3.1.3 ACO and ACO&GDA Method Performances

Table 3.3 show the fitness of the best tour found by ACO and ACO&GDA algorithms.

Table 3.3: ACO and ACO&GDA best fitness

| **Tsp problem** | | **ACO** | **ACO&GDA** |
|---|---|---|---|
| **att48** | 10628 | 12012 | **10628** |
| **att532** | 27686 | 32589 | **28321** |
| **berlin52** | 7542 | 8092 | **7657** |
| **Lin105** | 14379 | 16213 | **14434** |
| **Ch150** | 6528 | 6871 | **6554** |
| **Eil51** | 426 | 472 | **426** |
| **Eil76** | 538 | 580 | **543** |

| | | | |
|---|---|---|---|
| **Eil101** | 629 | 746 | **630** |
| **fl417** | 11861 | 13443 | **12038** |
| **kroA100** | 21282 | 24698 | **21296** |
| **kroA150** | 26524 | 30669 | **26751** |
| **kroA200** | 29368 | 34543 | **29591** |
| **kroB100** | 22141 | 25856 | **22258** |
| **kroB150** | 26130 | 30320 | **26292** |
| **kroC100** | 20749 | 23342 | **20798** |
| **kroD100** | 21294 | 24465 | **21395** |
| **Lin318** | 42029 | 48250 | **42670** |
| **Pr76** | 108159 | 124032 | **108159** |
| **Pr107** | 44303 | 46389 | **44303** |
| **Pr124** | 59030 | 65145 | **59087** |
| **Pr136** | 96772 | 111739 | **102323** |
| **Pr144** | 58537 | 59553 | **58636** |
| **Pr152** | 73682 | 78784 | **73880** |
| **Rat99** | 1211 | 1437 | **1221** |
| **Rd100** | 7910 | 9386 | **7951** |
| **St70** | 675 | 742 | **677** |
| **Tsp225** | 3916 | 4578 | **3942** |

### 3.1.4 U-TACO Results

U-TACO which uses UAnt to make an initialization to pheromone trail matrix, the pheromone trail matrix later uses by Ant to find a better solution for TSP problem. UAnt can have partial tour of different length, here we tried three different value for tour length *1/4, 1/2,* and *3/4 of city numbers* for different TSP problem. The results obtained in 50 iterations for UAnt to initialize pheromone trail matrix and 20 iterations for ACO to construct a complete tour. The affect of variant in UAnt tour length for different TSP problem are shown in Figure 14, for *Lin105* TSP problem, Figure 15, for *Pr107* TSP problem, and Figure 16, for *Eil51* TSP problem.

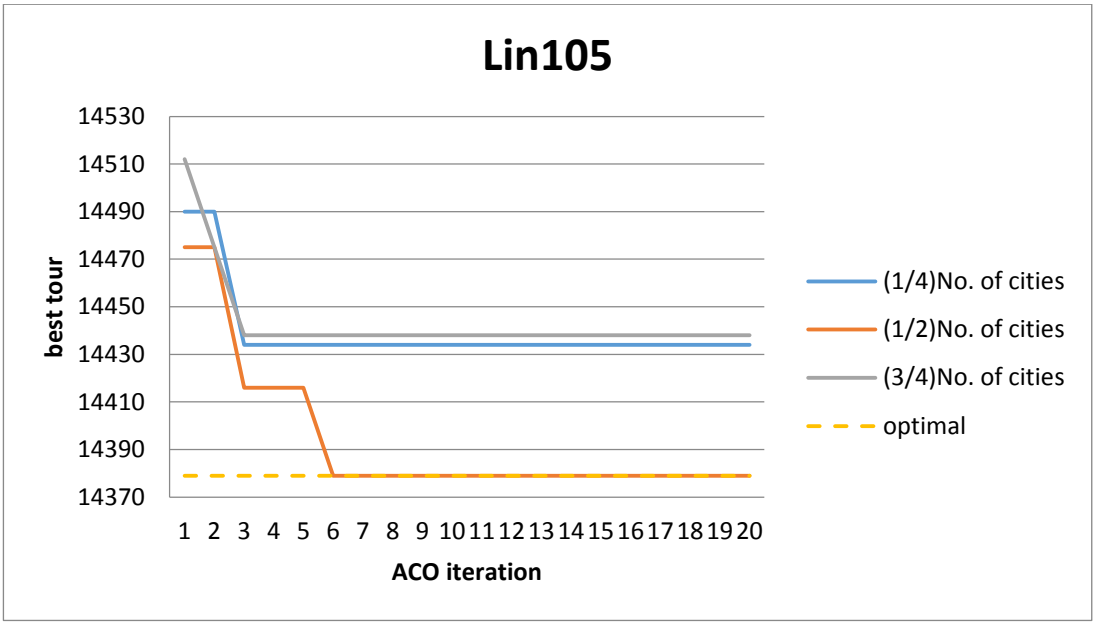


Figure 14: U-TACO for Lin105 over length 1/4, 1/2, and 3/4 of city numbers

For *Lin105* TSP problem we get tour fitness equal to 14434 for *1/4 * number of cities*, 14379 for *1/2 * number of cities*, and 14438 for *3/4 * number of cities.*

U-TACO find the optimal value for *Lin105* which is equal to 14379 at iteration 6 over 20 iteration, only when UAnt had make partial tour of length equal to *1/2 * number of cities.*

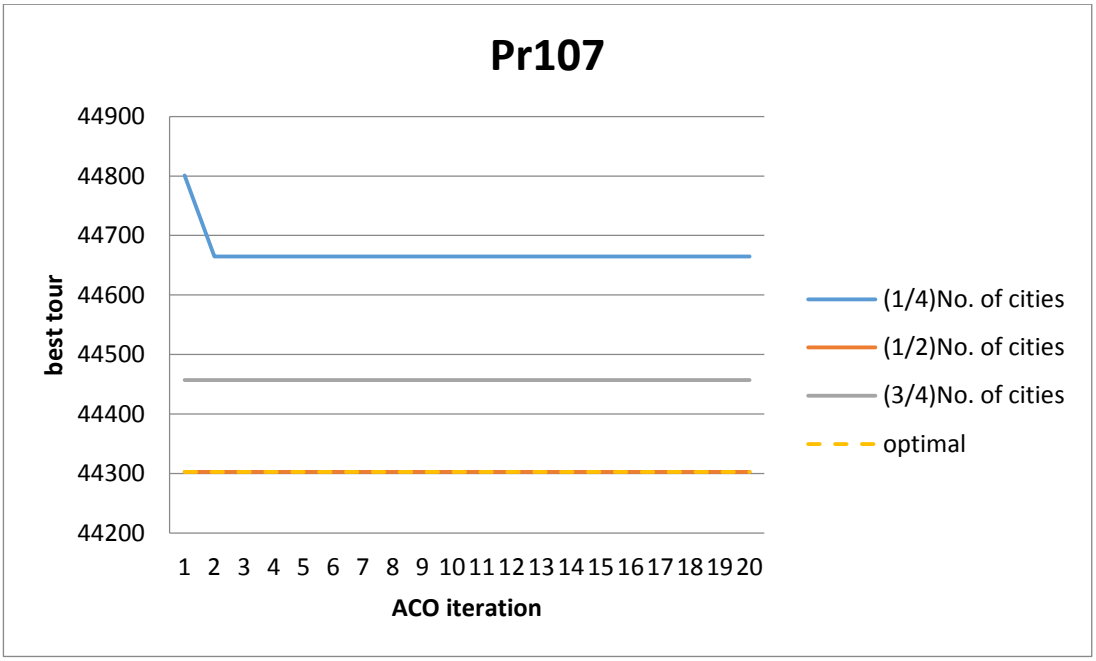


Figure 15: U-TACO for Pr107 over length 1/4, 1/2, and 3/4of city numbers

For *Pr107* TSP problem we get tour fitness equal to 44665 for *1/4 * number of cities,* 44303 for *1/2 * number of cities*, and 44457 *for 3/4 * number of cities*.

U-TACO finds the optimal value for *Pr107* which is equal to 44303 at iteration 1 over 20 iteration, only when UAnt had makes partial tour of length *1/2 * number of cities.*

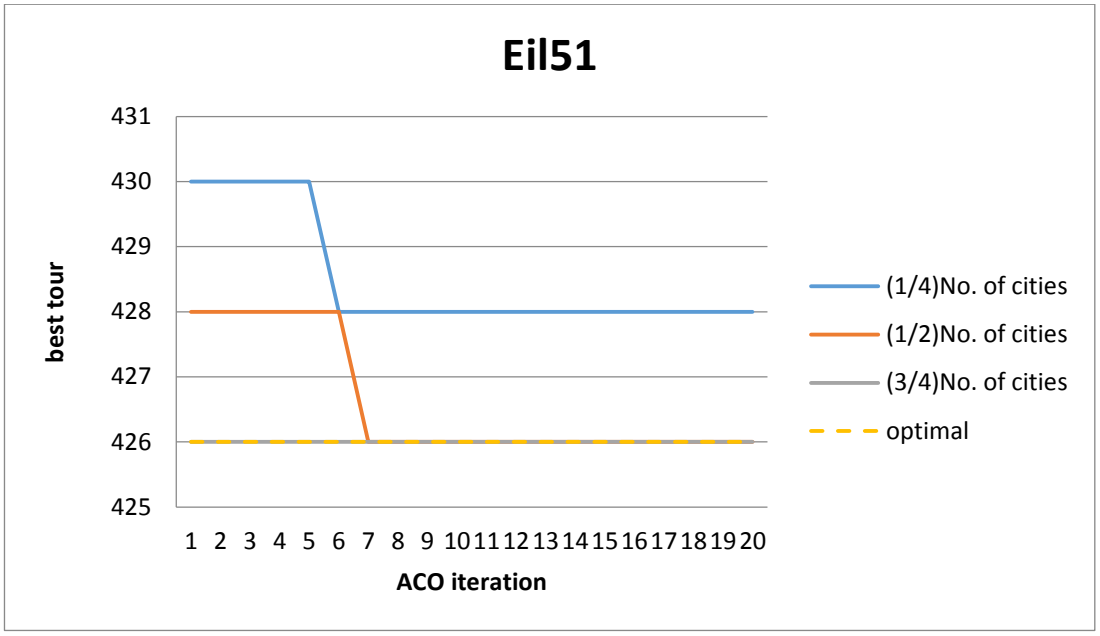


Figure 16: U-TACO for Eil51 over length 1/4, 1/2, and 3/4 of city numbers

For *Eil51* TSP problem we get tour fitness equal to 428 for *1/4 * number of cities*, 426 for *1/2 * number of cities*, and 426 for *3/4 * number of cities*.

U-TACO find the optimal value for *Eil51* which is equal to 426 at iteration 1 over 20 iteration, when UAnt had make partial tour of length *3/4 of number of cities*, and at iteration 7 over 20 iteration when UAnt had make partial tour of length *1/2 * number of cities.*

From Figure 14, Figure 15, and Figure 16, we concluded that U-TACO have a best result when UAnt makes a partial tour of length *1/2 * number of cities.*

Table 3.4 shows the U-TACO results when UAnt had make a partial tour of length *1/2 * number of cities* for the same collection of TSP problems that have been used before in testing (ACO and ACO&GDA) the optimal value for 6 TSP problems are founded.

Table 3.4: U-TACO results with 20 iterations

| Tsp problem | Optimal Tour | U-Turning ACO | Time | # of function Evaluation | Optimal foundation |
|---|---|---|---|---|---|
| **att48** | 10628 | 10628 | 72.608541 | 210960 | Iteration=2<br>Time=56.832731<br>Evaluation= 21096 |
| **att532** | 27686 | 28218 | 43566.033S | 2338140 | Not |
| **berlin52** | 7542 | 7542 | 92.900329s | 228540 | Iteration=3<br>Time=76.081437s<br>Evaluation=34281 |
| **Lin105** | 14379 | 14379 | 467.574097s | 461475 | Iteration=1<br>Time=358.475811s<br>Evaluation=23073 |
| **Ch150** | 6528 | 6547 | 1368.303153s | 659250 | Not |
| **Eil51** | 426 | 427 | 75.529904s | 224145 | Not |
| **Eil76** | 538 | 538 | 349.082318s | 334020 | Iteration=7<br>Time=313.814483s<br>Evaluation=116907 |
| **Eil101** | 629 | 629 | 439.032526s | 443895 | Iteration=8<br>Time=367.8602123 s<br>Evaluation=177558 |
| **fl417** | 11861 | 12045 | 20596.70082s | 1832715 | Not |
| **kroA100** | 21282 | 21282 | 415.284487s | 439500 | Iteration=6<br>Time=344.404586s<br>Evaluation=131850 |
| **kroA150** | 26524 | 26618 | 1243.611930s | 659250 | Not |
| **kroA200** | 29368 | 29472 | 2808.580653s | 879000 | Not |
| **kroB100** | 22141 | 22200 | 439.673347s | 439500 | Not |
| **kroB150** | 26130 | 26242 | 1333.134116s | 659250 | Not |
| **kroC100** | 20749 | 20798 | 446.163972s | 439500 | Not |
| **kroD100** | 21294 | 21455 | 434.844920s | 439500 | Not |
| **Lin318** | 42029 | 42749 | 14079.42749s | 1397610 | Not |
| **Pr76** | 108159 | 108487 | 202.229656s | 334020 | Not |

| | | | | | |
|---|---|---|---|---|---|
| **Pr107** | 44303 | 44303 | 524.576606s | 470265 | Iteration=1<br>Time=392.417070s<br>Evaluation=23513 |
| **Pr124** | 59030 | 59030 | 680.470220s | 544980 | Iteration=5<br>Time=545.294855s<br>Evaluation=136245 |
| **Pr136** | 96772 | 100702 | 884.805854s | 597720 | Not |
| **Pr144** | 58537 | 58669 | 1155.302216s | 632880 | Not |
| **Pr152** | 73682 | 73867 | 1550.821031s | 668040 | Not |
| **Rat99** | 1211 | 1213 | 981.875703s | 435105 | Not |
| **Rd100** | 7910 | 7935 | 479.834143s | 439500 | Not |
| **St70** | 675 | 679 | 212.419589s | 307650 | Not |
| **Tsp225** | 3916 | 3928 | 3400.762717s | 88500 | Not |

### 3.1.5 ACO, ACO&GDA and U-TACO Method Performances

Table 3.5 show the fitness of the best tour found by ACO, ACO&GDA and U-TACO algorithms.

Table 3.5: ACO, ACO&GDA and U-TACO best fitness

| Tsp problem | Optimal value | ACO | ACO&GDA | U-Turning ACO |
|---|---|---|---|---|
| **att48** | 10628 | 12012 | **10628** | **10628** |
| **att532** | 27686 | 32589 | 28321 | **28218** |
| **berlin52** | 7542 | 8092 | 7657 | **7542** |
| **Lin105** | 14379 | 16213 | 14434 | **14379** |
| **Ch150** | 6528 | 6871 | 6554 | **6547** |
| **Eil51** | 426 | 472 | **426** | 427 |
| **Eil76** | 538 | 580 | 543 | **538** |
| **Eil101** | 629 | 746 | 630 | **629** |
| **fl417** | 11861 | 13443 | **12038** | 12045 |
| **kroA100** | 21282 | 24698 | 21296 | **21282** |

| | | | | |
|---|---|---|---|---|
| **kroA150** | 26524 | 30669 | 26751 | **26618** |
| **kroA200** | 29368 | 34543 | 29591 | **29472** |
| **kroB100** | 22141 | 25856 | 22258 | **22200** |
| **kroB150** | 26130 | 30320 | 26292 | **26242** |
| **kroC100** | 20749 | 23342 | **20798** | **20798** |
| **kroD100** | 21294 | 24465 | 21395 | **21455** |
| **Lin318** | 42029 | 48250 | 42670 | **42749** |
| **Pr76** | 108159 | 124032 | **108159** | 108487 |
| **Pr107** | 44303 | 46389 | **44303** | **44303** |
| **Pr124** | 59030 | 65145 | 59087 | **59030** |
| **Pr136** | 96772 | 111739 | 102323 | **100702** |
| **Pr144** | 58537 | 59553 | **58636** | 58669 |
| **Pr152** | 73682 | 78784 | 73880 | **73867** |
| **Rat99** | 1211 | 1437 | 1221 | **1213** |
| **Rd100** | 7910 | 9386 | 7951 | **7935** |
| **St70** | 675 | 742 | **677** | 679 |
| **Tsp225** | 3916 | 4578 | 3942 | **3928** |

## 3.2 Comparison between ACO, ACO&GDA, and U-TACO Results with 20 ACO Iterations

ACO, ACO&GDA, and U-TACO had different results in time, function evaluation, and the foundation of optimal tour, this section present's some graphical charts that shows difference between ACO, ACO&GDA, and U-TACO. All the charts bellow represents the solutions of ACO, ACO&GDA, and U-TACO to a TSP problem with 20 ACO iterations.

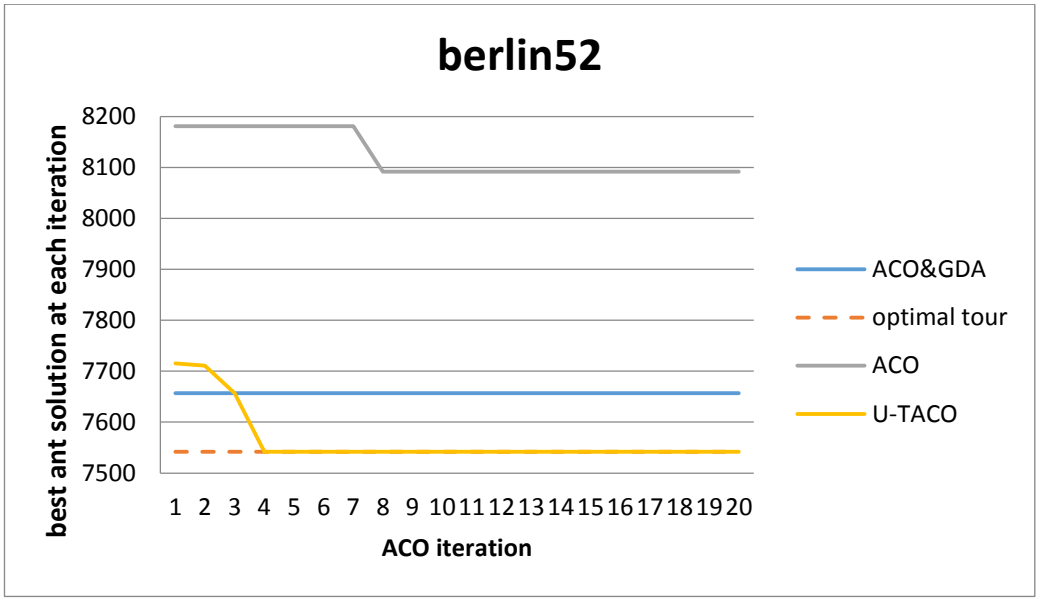


Figure 17: berlin52 results using ACO, ACO&GDA, and U-TACO

*Berlin52* is a symmetric TSP problem of type Euclidean distance two dimensions (*EUC_2*), size equal to 52 cities, and optimal tour fitness equal to 7542 units.

Figure 17, shows the results of all used algorithms in *berlin52* with 20 iterations, the best solution obtained by ACO was 8092 which is 550 units longer than optimal tour fitness of *berlin52*, and the best solution obtained by ACO&GDA was 7657 which is 115 units longer than optimal tour fitness, whereas the U-TACO reaches the optimal tour fitness value at iteration 3 with Time=76.081437seconds.

For *Berlin52* TSP problem, U-TACO is the best method between the used methods (ACO, ACO&GDA, and U-TACO) with respect to the fitness of best tour founded, which had finds the optimal tour.

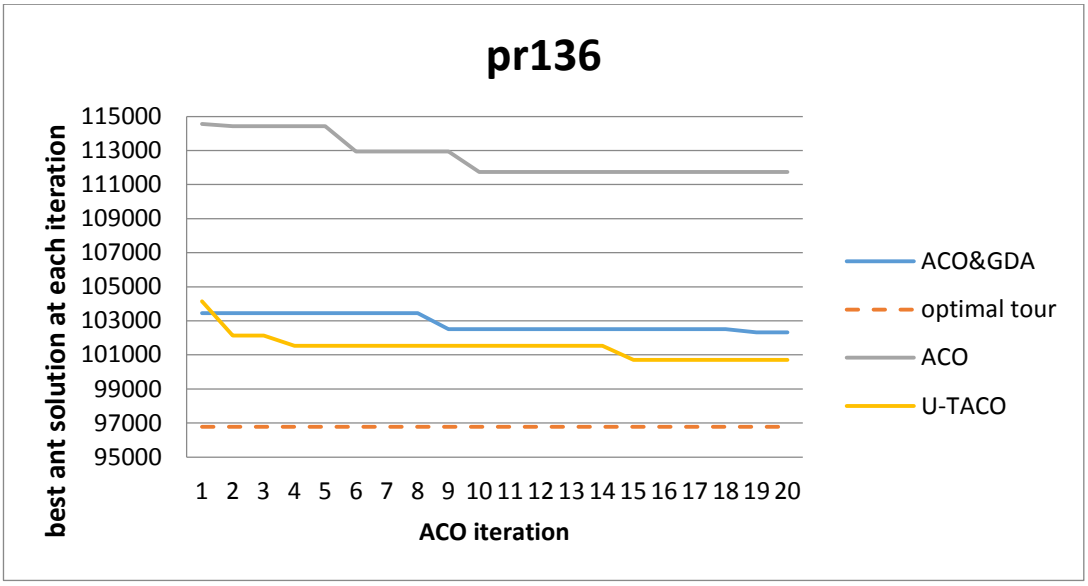


Figure 18: *Pr136* results using ACO, ACO&GDA, and U-TACO

*Pr136* is a symmetric TSP problem of type Euclidean distance two dimensions (*EUC_2*), size equal to 136 cities, and optimal tour fitness equal to 96772 units.

Figure 18, shows the results of all used algorithms in *Pr136* with 20 iterations, the best solution obtained by ACO was 111739 which is14967 units longer than optimal tour fitness of *Pr136*, whereas the best solution obtained by ACO&GDA was 102323 which is 5551 units longer than optimal tour fitness, and the best solution obtained by U-TACO was 100702 which is 3930 units longer than optimal tour fitness.

For *Pr136* TSP problem, U-TACO is the best method between the used methods (ACO, ACO&GDA, and U-TACO) with respect to the fitness of best tour founded.

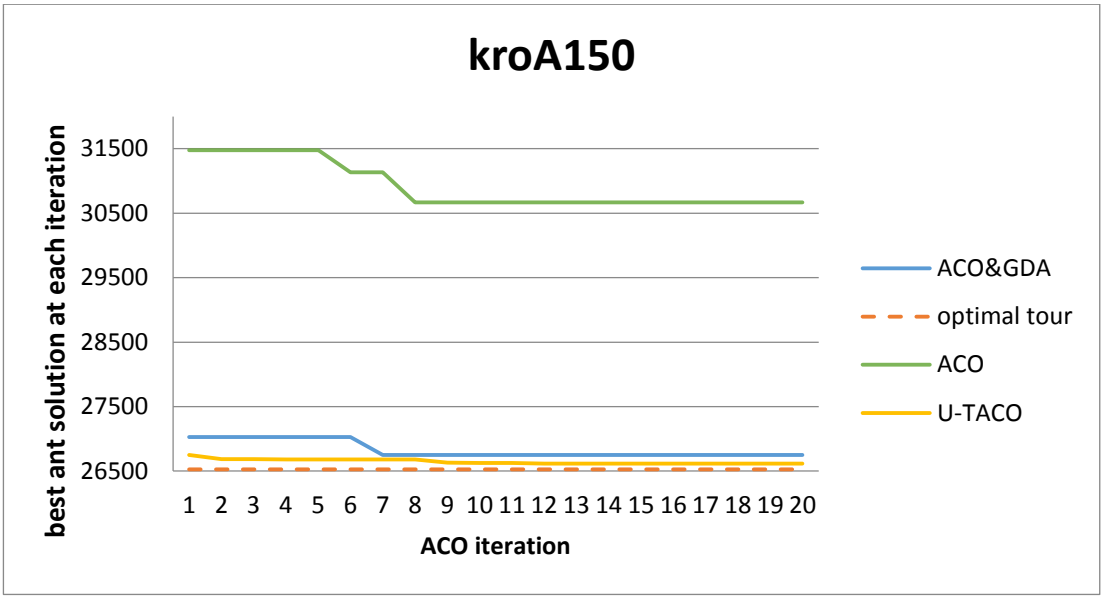

Figure 19: kroA150 results using ACO, ACO&GDA, and U-TACO

*KroA150* is a symmetric TSP problem of type Euclidean distance two dimensions (*EUC_2*), size equal to 150 cities, and optimal tour fitness equal to 26524 units.

Figure 19, shows the results of all used algorithms in *KroA150* with 20 iterations, the best solution obtained by ACO was 30669 which is 4145 units longer than optimal tour fitness of *KroA150*, whereas the best solution obtained by ACO&GDA was 26751 which is 227 units longer than optimal tour fitness, and the best solution obtained by U-TACO was 26618 which is 94 units longer than optimal tour fitness.

For *KroA150* TSP problem, U-TACO is the best method between the used methods (ACO, ACO&GDA, and U-TACO) with respect to the fitness of best tour founded.

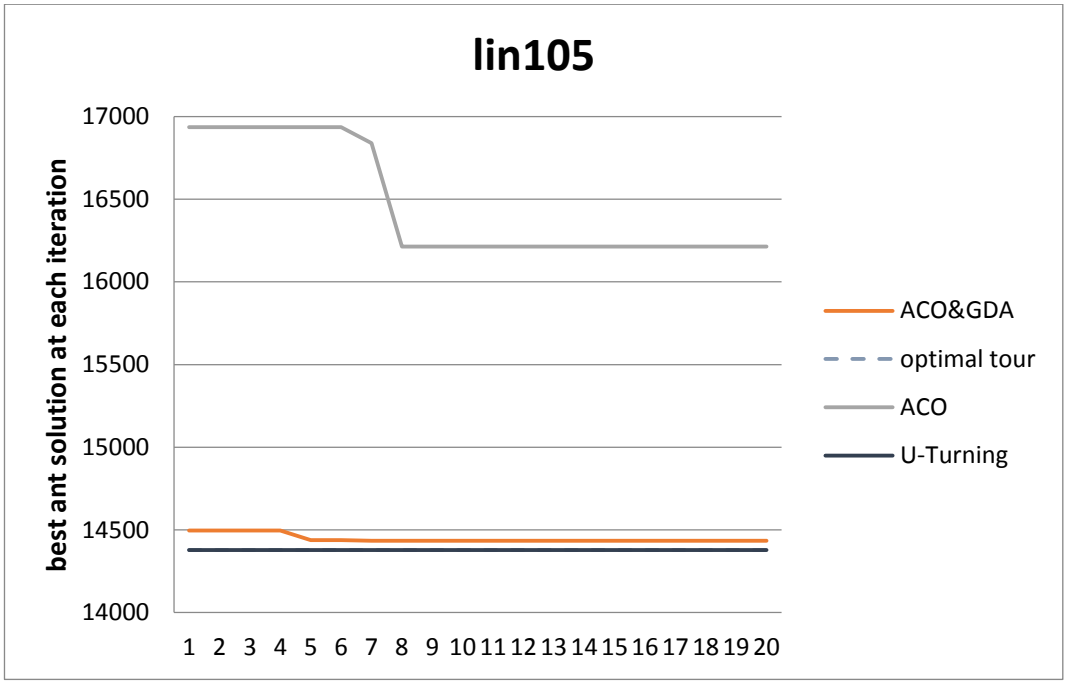

Figure 20: Lin105 results using ACO, ACO&GDA, and U-TACO

*Lin105* is a symmetric TSP problem of type Euclidean distance two dimensions (*EUC_2*), size equal to 105cities, and optimal tour fitness equal to 14379 units.

Figure 20, shows the results of all used algorithms in *Lin105* with 20 iterations, the best solution obtained by ACO was 16213 which is 1834 units longer than optimal tour fitness of *lin105*, and the best solution obtained by ACO&GDA was 14434 which is 55 units longer than optimal tour fitness, whereas the U-TACO reaches the optimal tour fitness value at iteration 1 with Time=358.475811 seconds.

For *Lin105*TSP problem, U-TACO is the best method between the used methods (ACO, ACO&GDA, and U-TACO) which had finds the optimal tour.

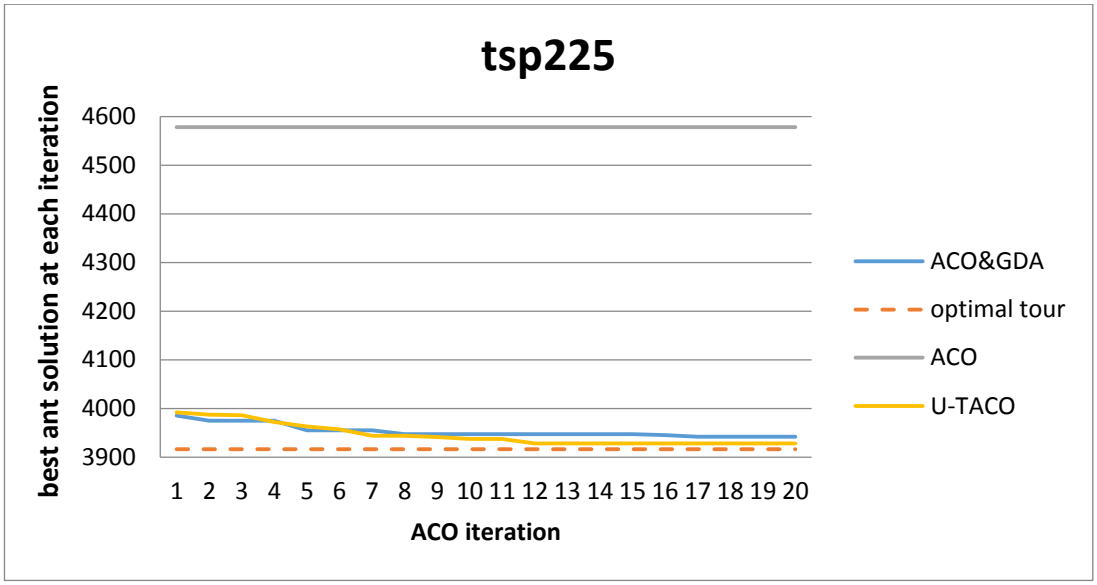


Figure 21: Tsp225 results using ACO, ACO&GDA, and U-TACO

*Tsp225* is a symmetric TSP problem of type Euclidean distance two dimensions (*EUC_2*), size equal to 225 cities, and optimal tour fitness equal to 3916 unit.

Figure 21, shows the results of all used algorithms in *Tsp225* with 20 iterations, the best solution obtained by ACO was 4578 which is 662 units longer than optimal tour fitness of *Tsp225*, whereas the best solution obtained by ACO&GDA was 3942 which is 26 units longer than optimal tour fitness, and the best solution obtained by U-TACO was 3928 which is 12 units longer than optimal tour fitness.

Again for *Tsp225*TSP problem, U-TACO is the best method between the used methods (ACO, ACO&GDA, and U-TACO) with respect to the fitness of best tour founded.

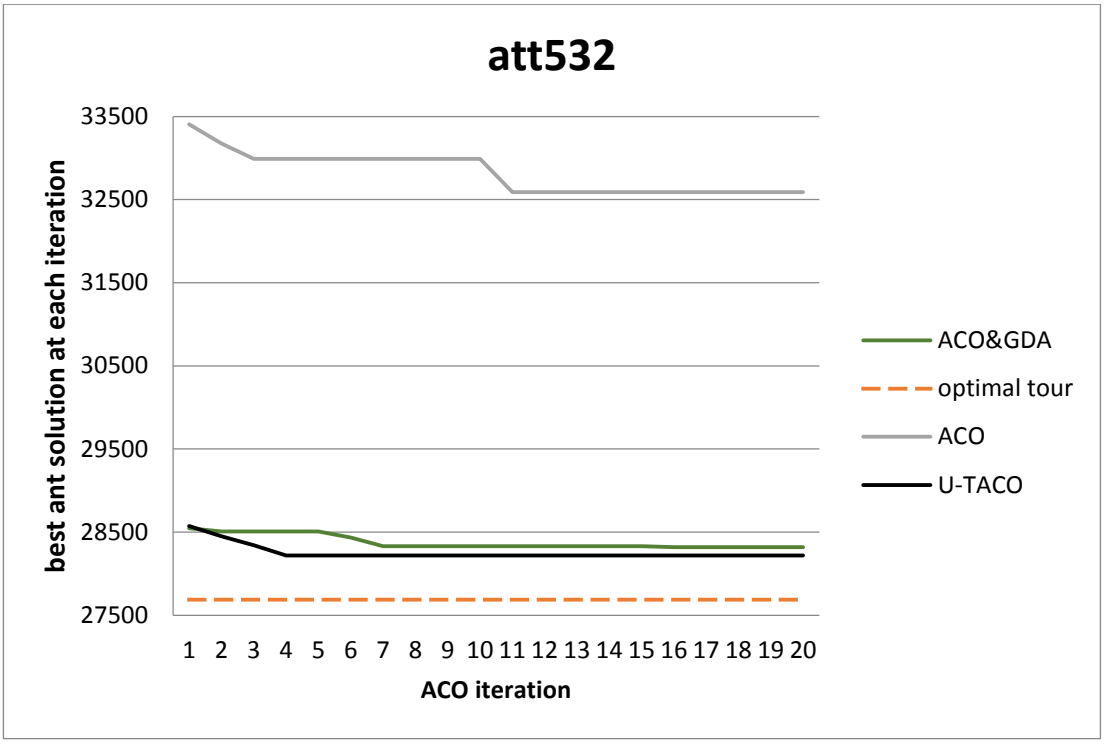


Figure 22: att532 results using ACO, ACO&GDA, and U-TACO

*Att532* is a symmetric TSP problem of type Euclidean distance two dimensions (*EUC_2*), size equal to 532 cities, and optimal tour fitness 27686 units.

Figure 22, shows the results of all used algorithms in *Att532* with 20 iterations, the best solution obtained by ACO was 32589 which is 4903 units longer than optimal tour fitness of *Att532*, whereas the best solution obtained by ACO&GDA was 28321 which is 635 units longer than optimal tour fitness, and the best solution obtained by U-TACO was 28218 which is 532 units longer than optimal tour fitness.

Again for *Att532* TSP problem, U-TACO is the best method between the used methods (ACO, ACO&GDA, and U-TACO) with respect to the fitness of best tour founded.

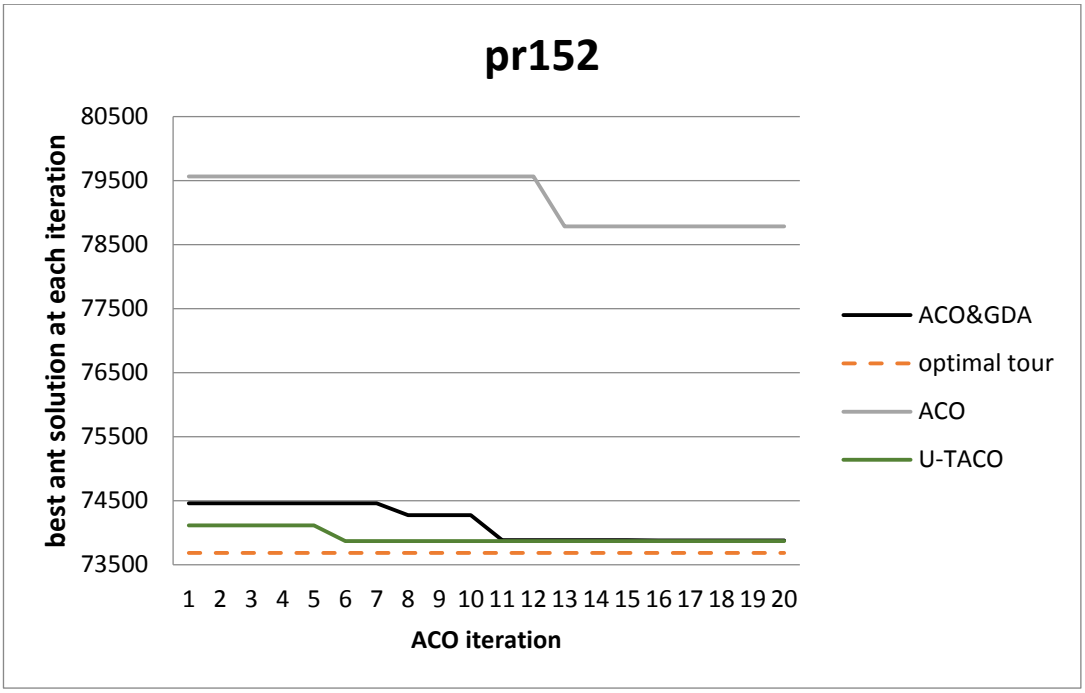

Figure 23: Pr152 results using ACO, ACO&GDA, and U-TACO

*Pr152* is a symmetric TSP problem of type Euclidean distance two dimensions (*EUC_2*), size equal to 152cities, and optimal tour fitness equal to 73682 units.

Figure 23, shows the results of all used algorithms in *Pr152* with 20 iterations, the best solution obtained by ACO was 78784 which is 5102 units longer than optimal tour fitness of *Pr152*, whereas the best solution obtained by ACO&GDA was 73880 which is 189 units longer than optimal tour fitness, and the best solution obtained by U-TACO was 73867 which is 185 unit longer than optimal tour fitness.

Again for *Pr152*TSP problem, U-TACO is the best method between the used methods (ACO, ACO&GDA, and U-TACO) with respect to the fitness of best tour founded.

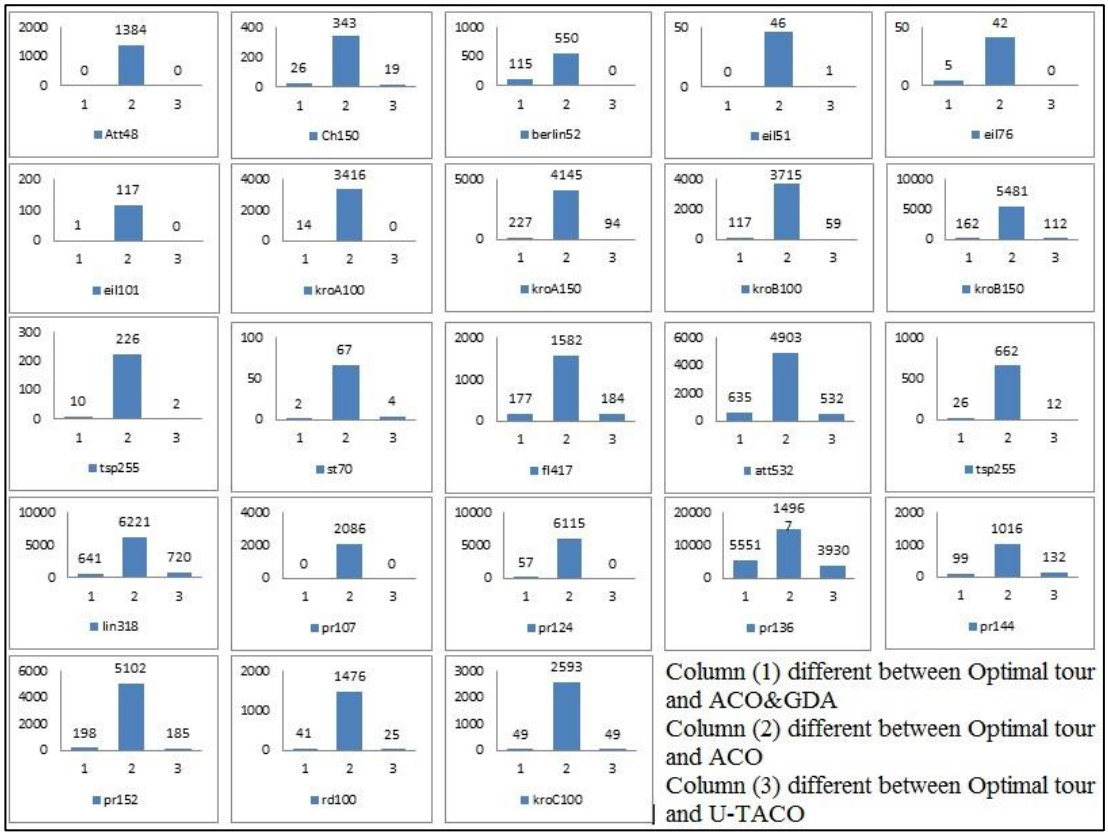


Figure 24: ACO, ACO&GDA, and U-TACO Results with 20 iterations

Figure 24 shows the difference in the fitness of the tours obtained by ACO, ACO&GDA, and U-TACO with 20 ACO iterations, where the first column of each chart represents the difference between the optimal tour fitness and the fitness of the best tour founded by ACO&GDA, which is a small quantity in almost all charts, shows the efficiency of ACO&GDA in constructing tours for various TSP problems, whereas the second column represents the difference between the optimal tour fitness and the fitness of the best tour founded by ACO, which is a large quantity in all charts, shows that ACO is inefficient with respect to ACO&GDA, and U-TACO for solving TSP problems. The last column represents the difference between the optimal tour fitness and the fitness of the best tour founded by U-TACO, which is the best in all charts.

In all the above charts we noted that the difference between optimal tour fitness and the best tour founded by ACO is large, and the difference decrease between optimal tour fitness and the best tour founded by ACO&DGA, whereas the difference between optimal tour fitness and the best tour founded by U-TACO is minimum.

## 3.3 Experimental results with 100 ACO iterations

This section shows the results of ACO, ACO&GDA, and U-TACO for 10 problems that the optimal tour could not be found with 20 ACO iterations by ACO, ACO&GDA, and U-TACO algorithms. All problems were solved with 100 ACO iterations and the parameter that determine the influence of pheromone trail $\alpha = 1$ and heuristic information $\beta = 5$, for each TSP problem number of ant equal to number of cities.

### 3.3.1 ACO Results

Table 3.6 shows the results of ACO algorithms. Here by using ACO with 100 iterations. It can be seen from this table that it is not possible to find any optimal solution for the given TSP problem from (TSPLIB95).

Table 3.6: ACO results with 100 iterations

| Tsp problem | Optimal Tour | ACO | Time | # of function Evaluation | Optimal foundation |
|---|---|---|---|---|---|
| **Eil51** | 426 | 472 | 127.366233s | 5100 | Not |
| **kroB100** | 22141 | 25774 | 559.368621s | 10000 | Not |
| **kroC100** | 20749 | 23074 | 577.269972s | 10000 | Not |
| **kroD100** | 21294 | 24354 | 566.206931s | 10000 | Not |
| **Pr76** | 108159 | 125166 | 167.635915s | 7600 | Not |
| **Pr144** | 58537 | 59176 | 570.405804s | 14400 | Not |
| **Pr152** | 73682 | 79564 | 443.512765s | 15200 | Not |
| **Rat99** | 1211 | 1356 | 200.324875s | 9900 | Not |

| | | | | | |
|---|---|---|---|---|---|
| **Rd100** | 7910 | 9386 | 449.774010s | 10000 | Not |
| **St70** | 675 | 740 | 193.447773s | 7000 | Not |

### 3.3.2 ACO&GDA Results

Table 3.7, shows the results of ACO&GDA algorithm, which have been run with 100 iterations on the same 10 TSP problem that we used before for ACO, the optimal tour for two TSP problem in total 10 TSP problem were found.

Table 3.7: ACO&GDA results with 100 iterations

| Tsp problem | Optimal Tour | GDA & ACO | Time | # of function Evaluation | Optimal foundation |
|---|---|---|---|---|---|
| **Eil51** | 426 | 426 | 92.695518s | 1025100 | Iteration=8<br>Time=9.426011s<br>Evaluation=8200 8 |
| **kroB100** | 22141 | 22220 | 534.16368s | 2010000 | Not |
| **kroC100** | 20749 | 20798 | 564.601225s | 2010000 | Not |
| **kroD100** | 21294 | 21395 | 526.477314s | 2010000 | Not |
| **Pr76** | 108159 | 108159 | 307.913820s | 1527600 | Iteration=20<br>Time=79.861093 s<br>Evaluation=3055 20 |
| **Pr144** | 58537 | 58636 | 1783.66841s | 2894400 | Not |
| **Pr152** | 73682 | 73818 | 2046.780874 | 3055200 | Not |
| **Rat99** | 1211 | 1221 | 534.826458s | 1989900 | Not |
| **Rd100** | 7910 | 7951 | 566.902234s | 2010000 | Not |
| **St70** | 675 | 677 | 218.909780s | 1407000 | Not |

### 3.3.3 ACO and ACO&GDA Method Performances

Table 3.8 show the fitness of the best tour found by ACO and ACO&GDA algorithms.

Table 3.8: ACO and ACO&GDA best fitness

| Tsp problem | Optimal Tour | ACO | GDA&ACO |
|---|---|---|---|
| **Eil51** | 426 | 472 | **426** |
| **kroB100** | 22141 | 25774 | **22220** |
| **kroC100** | 20749 | 23074 | **20798** |
| **kroD100** | 21294 | 24354 | **21395** |
| **Pr76** | 108159 | 125166 | **108159** |
| **Pr144** | 58537 | 59176 | **58636** |
| **Pr152** | 73682 | 79564 | **73818** |
| **Rat99** | 1211 | 1356 | **1221** |
| **Rd100** | 7910 | 9386 | **7951** |
| **St70** | 675 | 740 | **677** |

### 3.3.4 U-TACO Results

Table 3.9 shows the U-TACO results when UAnt had make a partial tour of length *1/2 * number of cities* for the same collection of TSP problems that have been used before in testing ACO and ACO&GDA we find optimal value for 7 in total 10 TSP problems.

Table 3.9: U-TACO results with 100 iterations

| Tsp problem | Optimal Tour | U-Turning ACO | Time | # of function Evaluation | Optimal foundation |
|---|---|---|---|---|---|
| **Eil51** | 426 | 426 | 162.245474 s | 1044225 | Iteration=20<br>Time=82.858010s<br>No.evaluation=20884 |
| **kroB100** | 22141 | 22141 | 909.405891 s | 2047500 | Iteration=75<br>Time=764.909430s<br>No.evaluation=153562 |
| **kroC100** | 20749 | 20749 | 927.863812 s | 2047500 | Iteration=4<br>Time=371.255152s<br>No.evaluation=81900 |
| **kroD100** | 21294 | 21398 | 1497.1134s | 2047500 | Not |
| **Pr76** | 108159 | 108159 | 439.465675 s | 1556100 | Iteration=22<br>Time=235.134446s<br>No.evaluation=342342 |
| **Pr144** | 58537 | 58636 | 2251.4319s | 2948400 | Not |
| **Pr152** | 73682 | 73682 | 2584.720247s | 3112200 | Iteration=97<br>Time=2534.341909s<br>No.evaluation=3018834 |
| **Rat99** | 1211 | 1211 | 940.880195 s | 2027025 | Iteration=55<br>Time=663.725786s<br>No.evaluation=1114863 |
| **Rd100** | 7910 | 7910 | 924.741646 s | 2047500 | Iteration=33<br>Time=536.255092s<br>No.evaluation=675675 |
| **St70** | 675 | 676 | 399.83319s | 1433250 | Not |

### 3.3.5 ACO, ACO&GDA and U-TACO Method Performances

Table 3.10 show the fitness of the best tour found by ACO, ACO&GDA and U-TACO algorithms.

Table 3.10: ACO, ACO&GDA and U-TACO best fitness

| Tsp problem | Optimal Tour | ACO | GDA&ACO | U-Turning ACO |
|---|---|---|---|---|
| **Eil51** | 426 | 472 | **426** | **426** |
| **kroB100** | 22141 | 25774 | 22220 | **22141** |
| **kroC100** | 20749 | 23074 | 20798 | **20749** |
| **kroD100** | 21294 | 24354 | **21395** | 21398 |
| **Pr76** | 108159 | 125166 | **108159** | **108159** |
| **Pr144** | 58537 | 59176 | **58636** | **58636** |
| **Pr152** | 73682 | 79564 | 73818 | **73682** |
| **Rat99** | 1211 | 1356 | 1221 | **1211** |
| **Rd100** | 7910 | 9386 | 7951 | **7910** |
| **St70** | 675 | 740 | 677 | **676** |

## 3.4 Comparison between ACO, ACO&GDA, and U-TACO Results with 100 ACO Iterations

ACO, ACO&GDA, and U-TACO had different results in time, function evaluation, and the foundation of optimal tour. U-TACO was the best algorithm used to solve TSP problem with in 20 ACO iterations. This section will present some graphical charts that show different in between ACO, ACO&GDA, and U-TACO for different TSP problems with 100 ACO iterations.

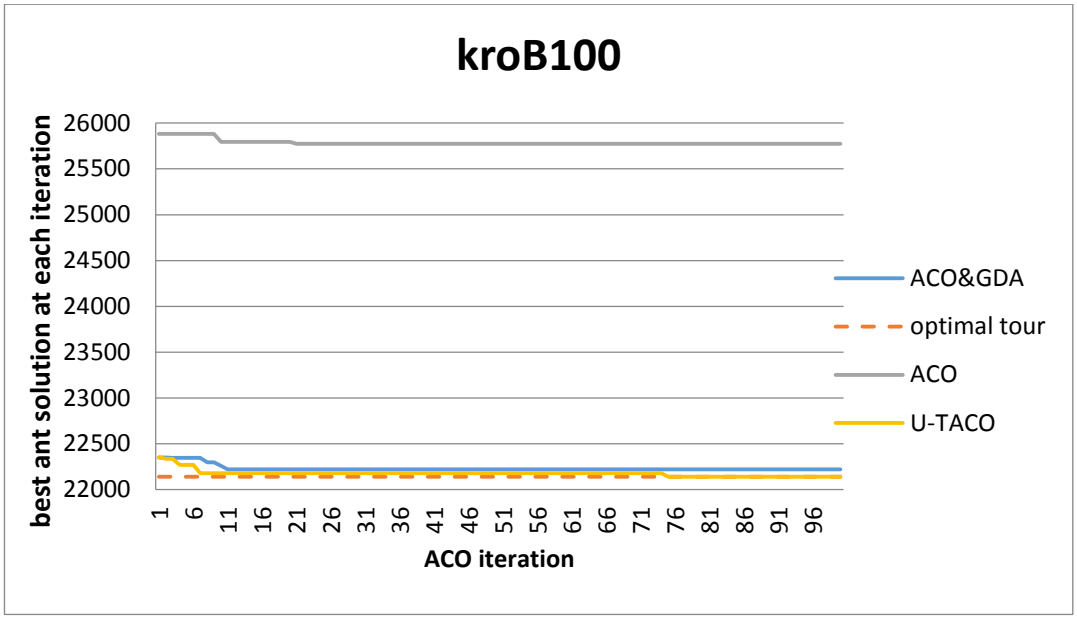

Figure 25: KroB100 results using ACO, ACO&GDA, and U-TACO

*KroB100* is a symmetric TSP problem of type Euclidean distance two dimensions (*EUC_2*), size equal to 100 cities, and optimal tour fitness equal to 22141 units.

Figure 25 shows the results of all used algorithms in *KroB100* with 100 iterations, the best solution obtained by ACO was 25774 which is 79 units longer than optimal tour fitness of *KroB100*, and the best solution obtained by ACO&GDA was 22220 which is 3633 units longer than optimal tour fitness, whereas the optimal solution found by U-TACO at iteration 75 with time equal 909.405891 seconds.

For *KroB100* TSP problem, U-TACO is the best method between the used methods (ACO, ACO&GDA, and U-TACO) with respect to the fitness of best tour founded.

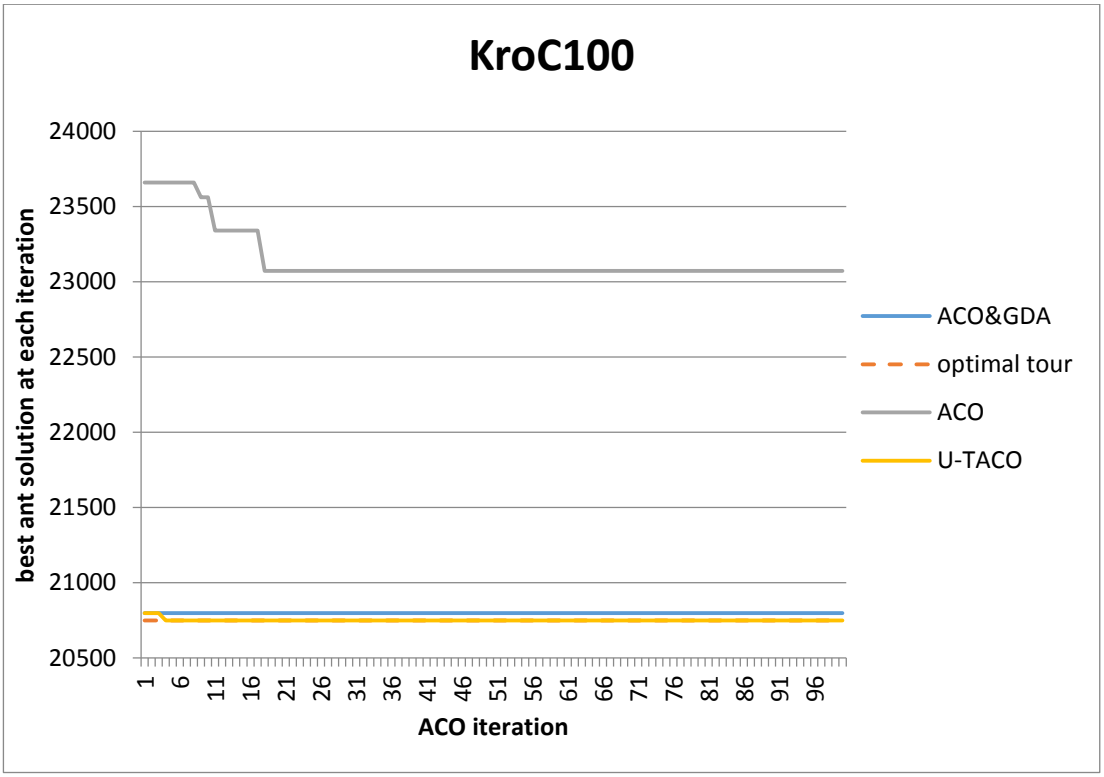


Figure 26: KroC100 results using ACO, ACO&GDA, and U-TACO

*KroC100* is a symmetric TSP problem of type Euclidean distance two dimensions (*EUC_2*), size equal to 100 cities, and optimal tour fitness equal to 20749 units.

Figure 26 shows the results of all used algorithms in *KroC100* with 100 iterations, the best solution obtained by ACO was 23074 which is 2325 units longer than optimal tour fitness of *KroC100*, and the best solution obtained by ACO&GDA was 20798 which is 49 units longer than optimal tour fitness, whereas the optimal solution found by U-TACO at iteration 4 with time equal 371.255152 seconds.

For *KroC100* TSP problem, U-TACO is the best method between the used methods (ACO, ACO&GDA, and U-TACO) with respect to the fitness of best tour founded.

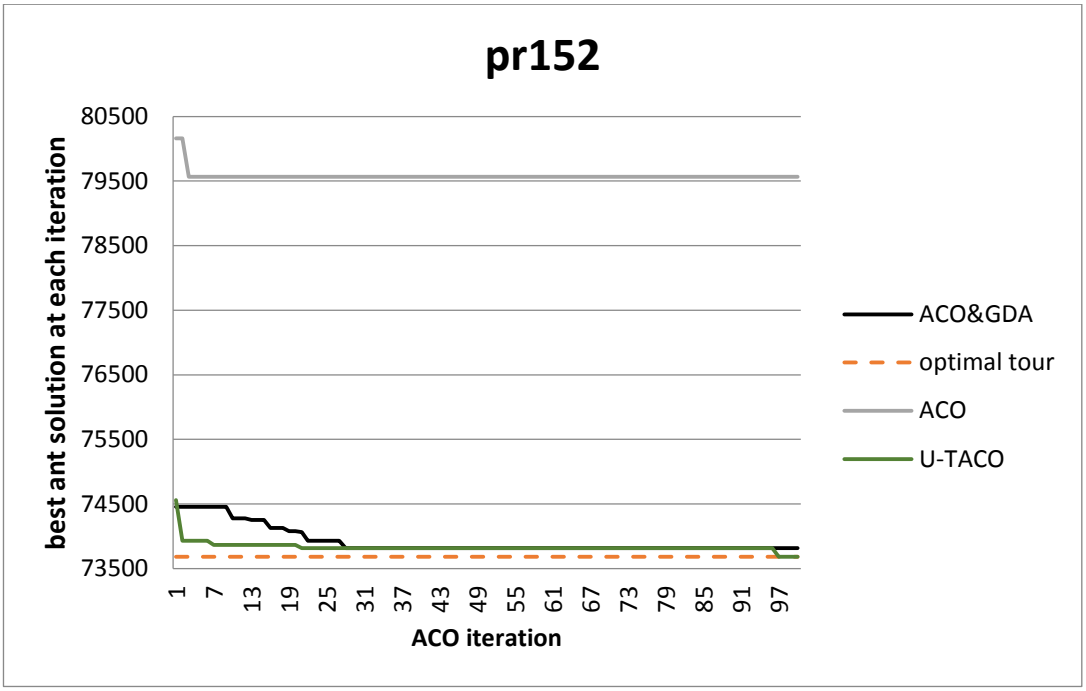

Figure 27: Pr152 results using ACO, ACO&GDA, and U-TACO

*Pr152* is a symmetric TSP problem of type Euclidean distance two dimensions (*EUC_2*), size equal to 100 cities, and optimal tour fitness equal to 73682 units.

Figure 27 shows the results of all used algorithms in *Pr152* with 100 iterations, the best solution obtained by ACO was 79564 which is 5882 units longer than optimal tour fitness of *Pr152*, and the best solution obtained by ACO&GDA was 73818 which is 136 units longer than optimal tour fitness, whereas the optimal solution found by U-TACO at iteration 97 with time equal 2534.341909 seconds.

For *Pr152* TSP problem, U-TACO is the best method between the used methods (ACO, ACO&GDA, and U-TACO) with respect to the fitness of best tour founded.

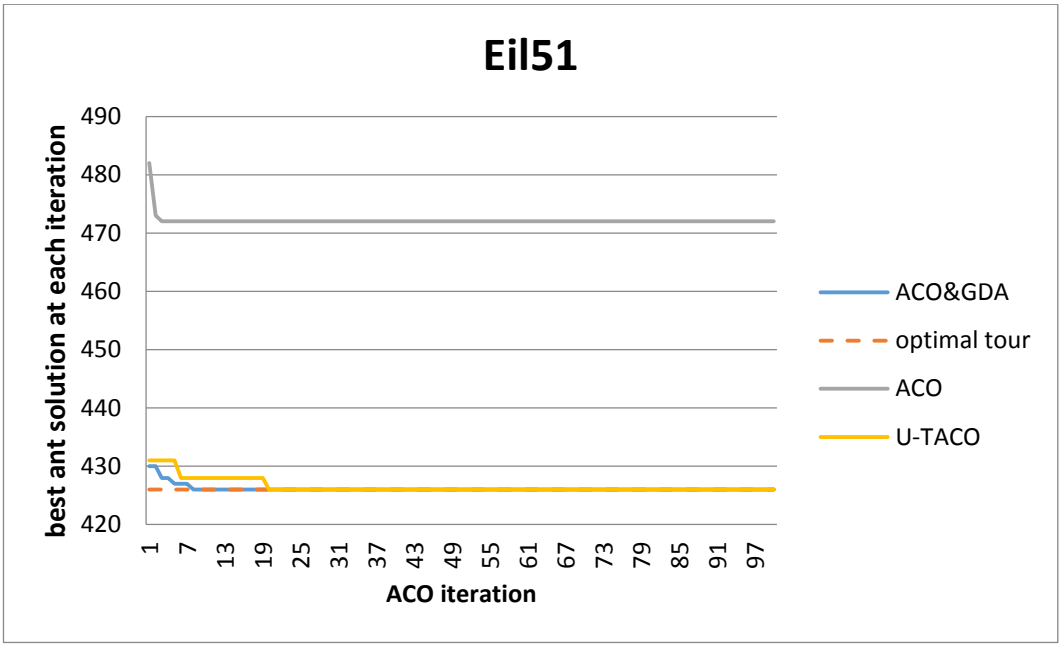

Figure 28: Eil51 results using ACO, ACO&GDA, and U-TACO

*Eil51* is a symmetric TSP problem of type Euclidean distance two dimensions (*EUC_2*), size equal to 51 cities, and optimal tour fitness equal to 426 units.

Figure 28, shows the results of all used algorithms in *Eil51* with 100 iterations, the best solution obtained by ACO was 472 which is 46 units longer than optimal tour fitness of *Eil51*, and the best solution obtained by ACO&GDA was 426 which is equal to optimal tour fitness, ACO&GDA found the optimal tour at iteration 8 with time equal 9.426011 seconds, whereas the U-TACO reaches the optimal tour fitness value at iteration 20 with Time=82.858010 seconds.

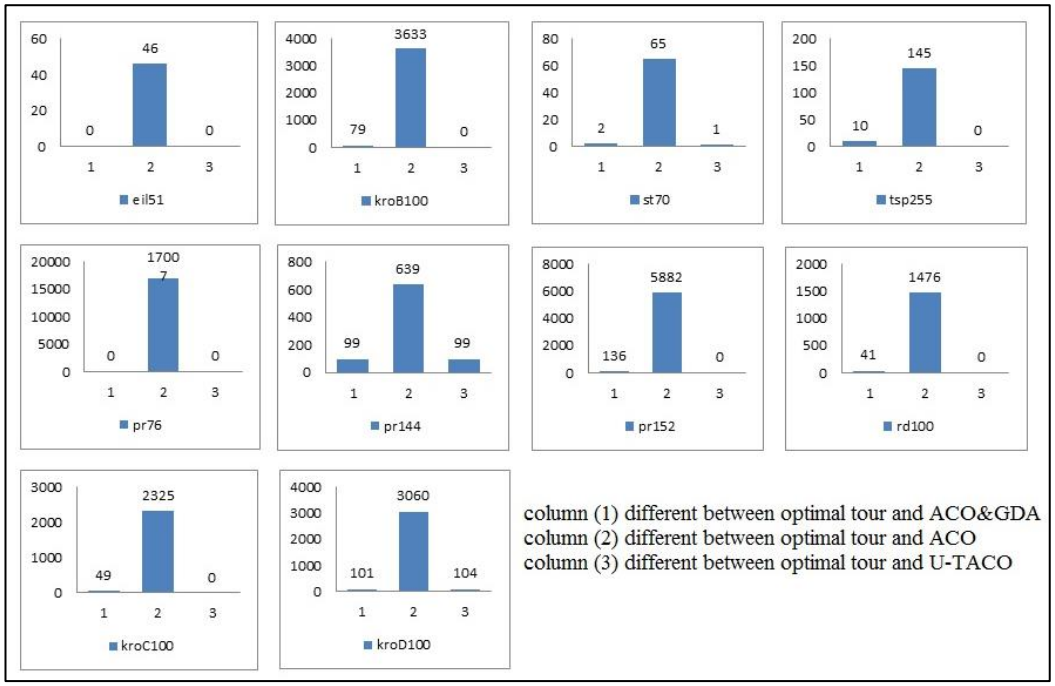


Figure 29: ACO, ACO&GDA, and U-TACO Results with 100 iterations

Figure 29, shows the different in the fitness of the tours obtained by ACO, ACO&GDA, and U-TACO with 100 ACO iterations, where The first column of each chart represents the difference between the optimal tour fitness and the fitness of the best tour founded by ACO&GDA, which is a small quantity in almost all charts, shows the efficiency of ACO&GDA in constructing tours for various TSP problems, whereas the second column represents the difference between the optimal tour fitness and the fitness of the best tour founded by ACO, which is a large quantity in all charts, shows that ACO is inefficient with respect to ACO&GDA, and U-TACO in solving TSP problems. The last column represents the difference between the optimal tour fitness and the fitness of the best tour founded by U-TACO, which is the best in all charts.

# Chapter 4

# CONCLUSION

Over the years Traveling Salesman Problem is one of the important optimization problem, which has been studied by many mathemation and computer scientist. In this study we considered different symmetric TSP (STSP) Problems from (TSPLIB95) of type Euclidean distance of 2-Dimensions, and ATT (ATT is a special distance function for problem ATT48 and ATT532) of various sizes we tried to solve them using different local search based algorithms that construct a near optimal tour.

In this thesis we use Ant Colony Optimization (ACO) algorithm, Ant Colony Optimization Powered by Great Deluge Algorithm (ACO&GDA), and finally we designed U-Turning Ant Colony Optimization (U-TACO) algorithm, all the three Algorithms have been run in a same collection of TSP problems, the obtained results of ACO, ACO&GDA, and U-TACO algorithms were various according to tour fitness, time elapsed, function evaluation, and foundation of optimality. The results of ACO, ACO&GDA, and U-TACO algorithms are given in details Table 3.1, Table 3.2, and Table 3.3, with 20 ACO iterations, and Table 3.4, Table 3.5, and Table 3.6, with 100 ACO iterations.

Ant Colony Optimization (ACO) algorithm is used to solve symmetric TSP problems over 20 ACO iterations and 100 ACO iterations, the results were far (very large different) from the optimal tours, In general ACO construct tours quickly and needs a

few Function evaluation calculations with respect to other algorithm used in finding a tour for the same TSP problems.

Ant Colony Optimization powered by Great Deluge Algorithm (ACO&GDA) is used to solve symmetric TSP problems over 20 ACO iterations and 100 ACO iterations, the results were relatively good (optimal / near optimal) with small difference between it and the optimal tour. But generally using ACO&GDA increase the time elapsed in solving the problem and the number of function evaluations used.

ACO&GDA is better than ACO in the foundation of optimal tour, and generally in construction of tour, but it is weaker according to elapsed time and number of function evaluations.

U-Turning Ant Colony Optimization powered by Great Deluge Algorithm (U-TACO) is used to solve symmetric TSP problems over 50 iterations for UAnts and 20 ACO iterations and 100 ACO iterations for Ants, the results were very good (optimal / near optimal) with a small difference between the optimal tour. But generally using U-TACO have a large time elapsed in solving the problem and the number of function evaluations used.

U-TACO is the best algorithm used between ACO, ACO&GDA, and U-TACO in the foundation of optimal tour, and generally in construction of tour, but it is weakest according to elapsed time and number of function evaluations.

# REFERENCES


Abdullah, S., & Burke, K. (2006). A multi-start large neighbourhood search approach with local search methods for examination timetabling. International Conference on Automated Planning and Scheduling, pp. 334-337.

Al-Milli, N. (2014). Hybrid Genetic Algorithm with Great Deluge To Solve Constrained Optimization Problems. Journal of Theoretical And Applied Information Technology, 59(2).

AppleGate, D., Bixby, R., Chvatal, V., & Cook. W. (1998). On the Solution of the Traveling Salesman Problems. Documenta Mathematica – Extra Volume ICM, chapter 3. 645-656.

Beckers, R., Deneubourg, L., & Goss, S. (1992). Trails and U-turns in the selection of the shortest path by the ant Lasius Niger. J. Theoretical Biology. 159, 397–415.

Beni, G., & Wang, J. (1989). Swarm intelligence in cellular robotic systems. In NATO Advanced Workshop on Robots and Biological Systems, Il Ciocco, Tuscany, Italy.

Bentley L. (1992). Fast Algorithms for Geometric Traveling Salesman Problems, ORSA J. Comput. vol. 4-4, 387-411.

Blum, C. (2005). Ant colony optimization Introduction and recent trends, Physics of Life Reviews, 2(4), 353-373.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. 35 (3). ACM Computing Surveys. 268–308.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, Oxford.

Chandra B., Karloff, H., & Tovey, C. (1994). New Results on the Old k-Opt Algorithm for the TSP. Proceedings of the fifth annual ACM-SIAM Symposium on Discrete Algorithms.150-159.

Colorni, A., Dorigo, M., & Maniezzo, V. (1991). Distributed optimization by ant colonies, in: Varela, F., Bourgine, P. (Eds.), First Eur. Conference Artificial Life. 134–142.

Croes, A. (1958). A Method for Solving Traveling-Salesman Problems. Operations Research. 5, 791-812.

Deneubourg, J.-L., Aron, S., Goss, S., & Pasteels, J.-M. (1990). The self-organizing exploratory pattern of the argentine ant. Journal of Insect Behaviour, 3,159–168.

Denis, D. (2004-2005). Implementation and Applications of Ant Colony Algorithm, Faculties Universitaires Notre-Dame de la Paix, Namur, Institute of Informatique Annee acad_emique

Devx.com.(2015). Four Heuristic Solutions to the Traveling Salesperson Problem : Page 3. [online] Available at: http://www.devx.com/dotnet/Article/33574/0/page/3 [Accessed 22 Jan. 2015].

Dipak, P., & Nishant, D. (4 Apr 2011). Hamiltonian cycle and TSP: A backtracking approach. International Journal on Computer Science and Engineering (IJCSE) Vol. 3.

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

Dorigo, M. (1998). ANTS' 98. From Ant Colonies to Artificial Ants : First International Workshop on Ant Colony Optimization, ANTS 98, Bruxelles, Belgique, octobre.

Dorigo, M., & Di Caro, G. (1999). The ant colony optimization meta-heuristic, in: New Ideas in Optimization, D. Come, M. Dorigo, & F. Glover, eds, McGraw-Hill, New York. 11-32.

Dorigo, M., & Gambardella, L. (1997). Ant colonies for the travelling salesman problem. Biosystems, 43(2), pp.73-81.

Dorigo, M., & Gambardella, M. (1997). Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem, IEEE Transactions on Evolutionary Computation, volume 1, numéro 1, 53-66.

Dorigo, M., & Stützle, T. (2003). The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances. International Series in Operations Research & Management Science.

Dorigo, M., & Stützle, T. (2004). Ant Colony Optimization. Cambridge, MA: MIT Press. 65-66.

Dorigo, M., Bonabeau, E., & Theraulaz., G. (2000). Ant algorithms and stigmergy, Future Gener. Comput. Syst., Vol. 16. 8, 851–871.

Dorigo, M., Caro, G., & Gambardella, L. (1999). Ant Algorithms for Discrete Optimization. Artificial Life, 5(2), pp.137-172.

Dorigo, M., Maniezzo, V., & Colorni. A. (1996). Ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 26 (1) 29–41.

Dueck, G. (1993). New optimization heuristics: The great deluge algorithm and the record-to-record travel. Journal of Computational Physics, 104: 86–92.

Garey, R. & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NPCompleteness. W. H. Freeman.

Geetha, R. & Umarani Srikanth, G. (2012). Ant Colony Optimization in Different Engineering Applications: An Overview. International Journal of Computer Applications, 49(17), pp.19-25.

Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. Computers and Operational Research. 5, 533-549.

Goss, S., Aron, S., Deneubourg, L., & Pasteels, M. (1989). Self-organized shortcuts in the argentine ant. Naturwissenschaften. 76, 579–581.

Goyal, S. (2010). A Survey on Travelling Salesman Problem.1-9.

Grasse´, P. (1959). La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp. La the´orie de la stigmergie:

Hochbaum, S. editor. (1997). Approximation Algorithms for NP-Hard Problems. PWS Publishing Company, Boston.

Hölldobler, B. & Wilson, O. (1990). The Ant. Cambridge, MA: Harvard University Press. 756.

Iwr.uni-heidelberg.de, (2014). Projects. [online] Available at: http://www.iwr.uni-heidelberg.de/groups/comopt/projects/index.html [Accessed 12 Nov. 2014].

Jackson, E., & Ratnieks. L. (2006). Communication in ants, Current Biology, Vol. 16, No. 15, pp. 570–574.

Jaddi, N., & Abdullah, S. (2013). Hybrid of genetic algorithm and great deluge algorithm for rough set attribute reduction. Turk J Elec Eng & Comp Sci, 21, pp.1737-1750.

Johnson, D. (1990 ).Local Optimization and the Traveling Salesman Problem, Proceedings of the 17th Colloquium on Automata Languages and Programming, Lecture Notes in Computer Science, 443, 446-461.

Johnson, D., & McGeoch, L. (2002). Experimental analysis of heuristics for the STSP. In Gutin and Punnen. chapter 9. 369–444.

Johnson, D., & Papadimitriou, C. (1985). Computational complexity. In LaLer et al. chapter 3, 37–86.

Karlson, P., & Lüscher, M. (1959) Pheromones: a new term for a class of biologically active substances. Nature. 183, 55–56.

Kennedy, J., Eberhart, G., & Shi, Y. (2001). Swarm Intelligence, Morgan Kaufmann, San Francisco, GA.

LaLer, L., Lenstra, K., Rinnooy K., & Shmoys. B. (1985) The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. John W., & Sons,.

Landa-Silva, D., & Obit, J.H. (2009). Evolutionary nonlinear great deluge for university course timetabling. International Conference on Hybrid Artifcial Intelligence systems.

Laporte, G. (1992). The Traveling Salesman Problem: An overview of exact and approximate algorithms. European Journal of Operational Research, 59, 231-247.

Lin, S. (1965). Computer Solutions of the Traveling-Salesman Problem. Bell Systems Technical Journal. 44, 2245-2269.

Lin, S., & Kernighan, W. (1973). An effective heuristic algorithm for the traveling salesman problem. Operations Research. 21, 498-516.

Louis J., & Gong L., (2000). Case injected genetic algorithms for traveling salesman problems, Information Sciences. 122, 201-225.

Mahalanobis, C. (1940). A sample survey of the acreage under jute in Bengal. Sankhyu.4, 511-530.

Merkle, D. & Middendorf, M. (2008). Swarm intelligence and signal processing [DSP Exploratory]. IEEE Signal Process. Mag., 25(6), pp.152-158.

Misevičius, A., Ostreika, A., Šimaitis, A. & Žilevičius, V. (2007). Improving local search for the traveling salesman problem. Issn 1392 – 124x information technology and control, 36(2).

Mitra, D., Romeo, F., & Sangiovanni-Vincentelli, A. (Dec. 1985). Convergence and finite-time behavior of simulated annealing. vol. 24, pp. 761–767.

Mute-net.sourceforge.net, (2015). MUTE: Simple, Anonymous File Sharing. [online] Available at: http://mute-net.sourceforge.net/howAnts.shtml [Accessed 20 Jan. 2015].

Panyam, S. (2011). Stochastic Algorithms : Part 2 - Clever Algorithms in Python. [online] Sai Panyam.NET. Available at: http://www.saipanyam.net/2011/06/stochastic-algorithms-2.html [Accessed 22 Jan. 2015].

Russell, S. & Norvig, P. (2003). Artificial intelligence. 1st ed. Upper Saddle River, NJ [u.a.]: Prentice Hall.

Sahni, S. & Gonzalez, T. (1976). P-Complete Approximation Problems. JACM, 23(3), pp.555-565.

Schrijver, A. (2005). on the history of combinatorial optimization (till 1960), handbook of discrete optimization, Elsevier, Amsterdam. 1-68.

Schrijver, A. (Springer 2003). Combinatorial Optimization: Polyhedra and Efficiency. Berlin.

Theorsociety.com, (2014). The OR Society: O.R. Methods - Heuristics - A brief History of the Travelling Salesman Problem. [online] Available at: http://www.theorsociety.com/Pages/ORMethods/Heuristics/articles/HistoryTSP.aspx [Accessed 3 Nov. 2014].

Wikipedia, (2014). Travelling salesman problem. [online] Available at: http://en.wikipedia.org/wiki/Travelling_salesman_problem [Accessed 3 Nov. 2014].

Yuri B. (November 2003). Time-Predefined and Trajectory-Based Search: Single and Multiobjective Approaches to Exam Timetabling. PhD Thesis submitted to the University of Nottingham.