DISCRETE WAVELET TRANSFORM-BASED ANT COLONY OPTIMIZATION FOR EDGE DETECTION

by

Aminu Muhammad

Submitted to the Institute of Graduate Studies in Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical-Electronic Engineering

Mevlana (Rumi) University

2014

DISCRETE WAVELET TRANSFORM-BASED ANT COLONY OPTIMIZATION FOR EDGE DETECTION

submitted by **Aminu Muhammad** in partial fulfillment of the requirements for the degree of Master of Science in Electrical and Electronic Engineering Department, Mevlana (Rumi) University

APPROVED BY:

Examining Committee Members:

Assist. Prof. Dr. Mohammad Shukri Salman . . . . . .

(Thesis Supervisor)

Assist. Prof. Dr. Nurdan Baykan . . . . . .

Assist. Prof. Dr. Alaa Eleyan . . . . . .

Prof. Dr. M. Uğur Ünver . . . . . .

Head of Department, Electrical-Electronic Engineering

Assoc. Prof. Dr. Ali Sebetci . . . . . .

Director, Institute of Graduate Studies in Science and Engineering

DATE OF APPROVAL (/ / 2014)

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Aminu Muhammad

Signature:

ABSTRACT

DISCRETE WAVELET TRANSFORM-BASED ANT COLONY OPTIMIZATION FOR EDGE DETECTION

Aminu Muhammad

M.Sc. Thesis, 2014

Thesis Supervisor: Assist. Prof. Dr. Mohammad Shukri Salman

Keywords: Ant Colony Optimization, Artificial Bee Colony Optimization, Genetic Algorithm, Simulated Annealing, Discrete-Wavelet Transform, Travelling Salesman Problem, Edge Detection.

Evolutionary Optimization has attracted many researchers to use it in solving many optimization problems that have no trivial solutions. Some of these techniques include; Genetic Algorithms (GA), Simulated Annealing (SA), Artificial Bee Colony (ABC), Ant Colony Optimization (ACO), etc.

In this thesis, we first compare the performance of GA, SA, ABC and ACO algorithms in solving the well-known Travelling Salesman Problem (TSP). From the results obtained, the ACO algorithm has shown significant performance compared to the others. Hence, the performance of the ACO algorithm is tested in the 2-Dimensional (2-D) case for edge detection.

In the last part of this work, the conventional 2-D ACO performance is tested in edge detection problem. It shows high performance. However, this performance can be improved further by transforming the input into different domain from the real time. Hence, we apply a Discrete-Wavelet Transform (DWT) at the input of the 2-D ACO algorithm which provides us denser and clearer images compared to the conventional ACO.

Simulations show that the proposed 2-D DWT-based ACO provides very high performance compared to the conventional one, especially, when the input image is buried with noise.

Dedicated to my Family and Friends

ACKNOWLEDGEMENTS

I am wholeheartedly thankful to my supervisor, Assist. Prof. Dr. Mohammad Shukri Salman for giving me an opportunity to work under his supervision. This work would not have been successful without his patience, guidance and encouragement. I cannot thank you enough for the unlimited support you have done to me sir!

Among the people, I would like to specially deliver my sincere gratitude to Assist. Prof. Dr. Alaa Eleyan for his support and good advices. In addition, my appreciations go to my parents, brothers and sisters who have been helping and assisting me since upbringing. It is an honor for me to offer my gratitude to the Kano State Scholarship Board and the entire people of Kano for giving me this magnificent opportunity to undergo for my master's degree program.

In addition, I wish to thank all my friends for their prayers and well-wishers, especially, Nigerians who are here with me at Mevlana University, Konya, Turkey for the same program.

Lastly and above all, I give all love, thanks, honors and glories to our creator, ALLAH the sustainer, the cherisher for making everything achievable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ ABBREVIATION

Symbol	Explanation
α	Pheromone trail or arc-fitness influence
β	Heuristic information influence
ΔE	Energy change
Δf	Objective function change
ζ	Arc-fitness value
η	Heuristic information
ξ	Chromosome solution
π	Permutation of cities
ρ	Evaporation coefficient
σ^2	Noise variance
τ	Pheromone trail
ϕ	$rand(-1,1)$
Φ	Arc-fitness matrix
ψ	Pheromone decay coefficient
ω	Preferred path
A	Set of allowed cities.
d	Distance
D	Dance duration
e	Decision tolerance value
F	Preferred city
K_S	Dance scaling factor
k_B	Boltzmann's constant

L	Tour length
N	Neighborhood cities
N_B	Number of detected edge pixels
N_I	Number of ideal edge pixels
p	Transitions probability
pf	Tour profitability
p_{cd}	Percentage of true positive edge pixels
$p_{fol.}$	Probability of selecting a bee's tour
p_{nd}	Percentage of false negative edge pixels
p_{wd}	Percentage of false positive edge pixels
T	Annealing temperature
T_h	Threshold value
V_c	Variation in intensity
w	Penalty constant
x	Food source position
z	Normalization factor
1-D	One dimensional
2-D	Two dimensional
ABC	Artificial bee colony
ACO	Ant Colony Optimization
cA_{j+1}	Approximation coefficient
$cD_{j+1}^{(h)}$	Decomposed horizontal coefficient
$cD_{j+1}^{(v)}$	Decomposed vertical coefficient
$cD_{j+1}^{(d)}$	Decomposed diagonal coefficient

EAs	Evolutionary algorithms
EP	Evolutionary programming
ES	Evolutionary strategy
FN	False negative
FP	False positive
FOM	Figure of merit
GA	Genetic algorithm
Hi_D	Higher decomposition pass filter
Hi_R	Higher reconstruction pass filter
Lo_D	Lower decomposition pass filter
Lo_R	Lower reconstruction pass filter
NP	Non-linear programming
SA	Simulated annealing
TP	True positive

CHAPTER ONE

LITERATURE REVIEW

1.1. Evolutionary Algorithms

Evolutionary Algorithms (EAs) have turned out to be well-known instruments for solving many complex optimization problems by the use of simulated Darwinian evolution processes [1]. In searching and optimization tasks, the use of EAs as a stochastic search method has been recently admired [2]. It has been applied as an all-purpose search method in areas like machine learning, process control, combinatorial optimization, etc. because of its parallelism, robustness and simplicity [3]. In EAs an amount of simulated individuals try to find out an optimal solution by exploring on the problem space to locate best areas [4]. The idea behind EAs is to generate children called offsprings during each iteration from the best part of the solutions by mutation or combination to produce even better solutions in the next generations [3]. An EA cycle can be shown in Fig. 1.1. The first stage is the initialization of individuals' population and is usually created randomly [5]. The fitness values which stand for the quality of the individuals are then calculated in the next stage. Parents are then selected from the individuals based on their fitness functions, the individuals with the higher quality are mostly selected to be the parents and the others are discarded. The parents selected are recombined to produce children called offsprings. These offsprings get certain characters of the parents they are made from. However, there are some evolutionary algorithms that do not use this method of recombination to produce new offsprings as we can see later in evolutionary strategy [5]. The children produced by recombination are then mutated in order to avoid having a solution that is similar to any of the solution in the initialization state. The mutation is done by adding a little bit randomness into the genes of the population. The next stage in the EA cycle after mutating the children produced is the evaluation. In this stage, the fitness values of the individuals recombined and mutated are computed using the objective function. The environmental pressure affects the individuals due to the environmental changes. Some of the individuals survive the environmental selection and some do not, this environmental selection usually eliminates the individuals with low quality. The ones with higher fitness values usually survive and become parents for the next generation. After the environmental

selection, the whole process is stopped if the termination criterion is met. Otherwise the next cycle starts.

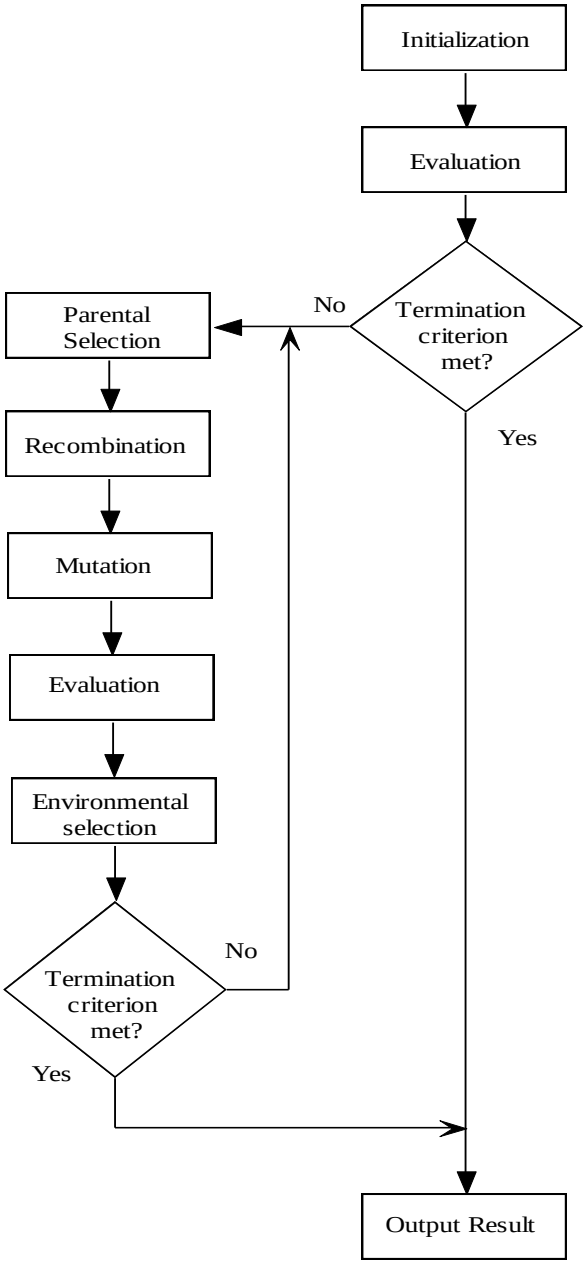


Figure 1.1: Evolutionary Algorithm Cycle, [5].

1.1.1. Paradigms

Evolutionary Algorithms (EAs) have many types, and they all differ a little bit but, in this thesis, we will be interested in three types, namely:

1. Genetic Algorithm
2. Evolution Strategy and
3. Evolutionary Programming

1.1.1.1. Genetic Algorithm

Genetic Algorithm (GA) was first introduced in United States by John Holland and his students in 1960s and 1970s [1]. GA is inspired by the natural selection theory of Charles Darwin based on biological evolution. In the field of artificial and adaptive systems, the use of genetic operators that includes; selection, crossover and mutation, and recombination as a basic of GA for solving problems were first used by John Holland [6]. Since the introduction of this algorithm, a lot of optimization problems like travelling salesman, data clustering, graph coloring, path planning in robotic system, etc. have been successfully tackled [6]. GA steps can be summarized as the following [7]:

1. Initialization: Here usually the initial populations of solution are generated randomly. The population can be large or small depending on the size needed.
2. Evaluation: Each individual of the population is then evaluated and its corresponding fitness is computed.
3. Selection: This process involves selecting the best individuals and discarding the bad ones. The selection is made based on the fitness values. The individuals with higher fitness values are more probable to be selected than the ones with low fitnesses. The fitter individuals selected are used for the next subsequent generations.
4. Crossover: This stage involves crossing the selected individuals (called parents) to form new individuals (called children). The impression behind crossover is that, the children produced inherit the best bits of the individuals. These children are then added to the populations. Crossover increases, in average, the quality of the solution [8].
5. Mutation: In order to avoid repetition of a solution, a little bit randomness is added into the genes of the populations otherwise each solution produced will be in the initial population. Mutation helps the algorithms to find out new states and this reduces the chances of converging to local optima [8].
6. Repeat: The process is repeated from step two until termination criterion is met.

GA has many advantages over other conventional optimization algorithms. Some of these important advantages include; parallelism and its ability to handle complex optimization

problems. Despite its ability to handle optimization problems where the objective fitness function is either; linear, continuous, stationary, non-linear, discontinuous, non-stationary, etc. [9], GA it still has some disadvantages in selecting the right parameters for the algorithms like rate of crossover and mutation, fitness function population, etc. It needs careful selection of these parameters for the algorithm to converge [6].

1.1.1.2. Evolution Strategy

Evolution Strategy (ES) was first established by Ingo Rechenberg, Hans-Paul Schwefel and other co-workers in the late 1960s and early 1970s in Germany [1]. Traditionally, ES was invented for parameter optimization problems [4]. As from the name implies, this algorithm is also inspired from the Darwinian evolution of natural selection [10]. In ES, not like GA, new individuals are generated without using crossover or any method alike, but instead a number of samples with the higher fitness values are selected. The samples with higher fitness values are selected and others with less are discarded. The successful of these individuals are used for the subsequent generation as parents, and the process continues. ES algorithms are simple, heuristic naturally and show very good performance [10]. ES usually uses a very low size of population $(1-20)$ unlike GA and is normally put in use to real-valued optimization problems [1].

In General, ES procedure can be summarized as follows:

1. Initialization: Here we initialize a population of individuals and each of the individuals is associated with an object parameter, a solution and a fitness value. However, a population in some instances may only contain one individual [11].
2. Selection: After the populations are initialized, one or more individuals called parents are selected and are duplicated and recombined to produce new individuals of population called children or offsprings.
3. Mutation: The children called offsprings produced in step 2 are then allowed to go through mutation and afterward become new population members.
4. The original size of the population is regained by the environmental selection [11].

1.1.1.3. Evolutionary Programming

Evolutionary Programming (EP) also aims to attain intelligent behavior by means of simulated evolution and fits for real-valued function and combinatorial optimization [12]. This form of EAs was initially introduced in the United States in the year 1960 by Laurence

J. Fogel for the evolution of finite state machine [1]. In Fogel's work, mutation operators play very vital role in alternating the finite state machines that were being evolved for a particular job [1]. In EP no recombination is applied among the individuals, every individual of the population precisely produce one offspring through mutation, so the number of parents is equal to the number of children. The children combined with the parents produce the total number of individuals, and the successful individuals are selected based on a probabilistic tournament [13].

1.2. Metaheuristic Algorithms

Metaheuristic Algorithms (MAs) are mostly nature-simulated algorithms as they are usually based on Darwin's theory of evolution [6]. In MAs, selection of solution and randomization are the two main important parts. Enough randomization helps the algorithm to keep away from being trapped into the local optima as it will search many minima including the global optima. Selection of the best solution guarantees the convergence of the system to the global minima [6].

1.2.1. Classifications of Metaheuristic Algorithms

Metaheuristic Algorithms can be classified depending on which criterion is taken into consideration in classifying them. Below are some of the major ways of classifying them.

1. Nature-inspired and non-nature inspired: Some of the metaheuristic algorithms are derived from nature and some are not. Bee Colony Algorithms, Ant Colony Algorithms and Genetic Algorithms are some of the algorithms derived from nature and hence classified as nature inspired algorithms. On the other hand, Tabu Search is a non-nature inspired algorithm. However, to some researches, this method of classification is not very vital in the sense that some algorithms hardly be put in one of the classes or can be put in both [14].
2. Population-based and Trajectory-based: Population-based algorithms are algorithms that operate on a population of solution at any time. Genetic Algorithm and Particle Swarm Optimization are good example of population based algorithms as they use a set of strings and multiple agents or particles, respectively. On the other hand, Trajectory-based algorithms work on a single solution at any time and includes; Simulated Annealing, Tabu Search, etc. [6].

3. Memory and Memory-less: Some metaheuristic algorithms have memory and make use of their search history and some have no memory. The memory-less algorithms only apply the current knowledge of the search process in order to ascertain the subsequent action. They do Markov Chain process [14].

1.2.2. Major Metaheuristic Algorithms

Since the advent of EAs around 1970s, a lot of metaheuristic algorithms are being introduced in order to overcome the shortcomings associated with the present numerical algorithms in finding the solution of difficult optimization problems [15]. Major algorithms include GA, ant colony algorithms (ACO), artificial bee colony algorithm (ABC), simulated annealing (SA), harmony search (HS), firefly algorithm (FA), particle swarm optimization (PSO), and so on. In this work, four of these metaheuristic algorithms which include; GA, SA, ABC and ACO will be discussed in the remaining part of this thesis and the process of their implementation in 1-D setting will be presented later in the report.

1.2.2.1. Genetic Algorithm

GA is inspired by the natural selection theory of Charles Darwin based on biological evolution. In the field of artificial and adaptive systems, the use of genetic operators that includes; selection, crossover and mutation, and recombination as a basic of GA for solving problems was first used by John Holland [6]. GA has been used in solving many optimization problems [6]. The whole ideas behind GA involve encoding the optimization functions to signify chromosomes, maneuver these chromosomes which can be either bits arrays or character arrays by the use of GA operators, and uses selection techniques to obtain good solutions to the optimization problems. However, after a termination criteria is met the final result is decoded to obtain the result.

In GA, good choice of fitness function is very vital for deciding the selection criterion to be utilized. In GA for function minimization, fitness values can be obtained using Eq. (1.1).

$$F = A - f(x), \tag{1.1}$$

where A is a constant value that leads F to be positive if selected to be greater than $f(x)$. Usually, if A is selected to be zero then, the aim is to maximize $f(x)$ and afterwards minimize the function F. Fitness functions are not only defined in this way, there are many

other ways [6]. One frequently used of them is to use individual fitness in relation to the total population given in Eq. (1.2).

$$F(x_i) = \frac{f(\xi_i)}{\sum_{i=1}^{N} f(\xi_i)}, \tag{1.2}$$

where ξ_i represents the individual chromosome solutions and N is the total population size [6].

Crossover and mutation are two important operators. Crossover is having the higher probability and is done by swapping the segments positions of the two chromosomes interchangeably at random. Crossover can be at various points in a chromosome. Fig. 1.2 shows a single point crossover [6].

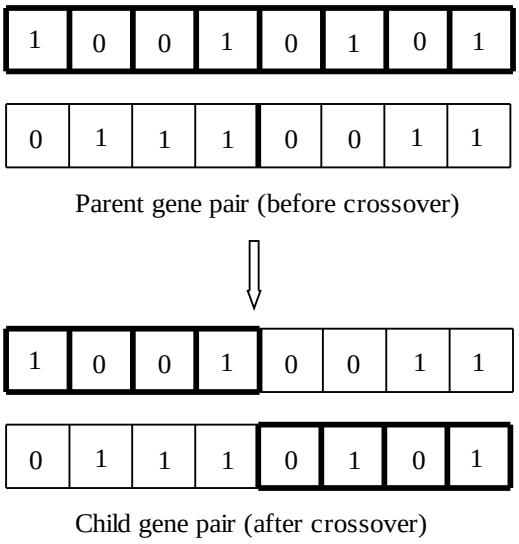


Figure 1.2: Genetic Algorithm single-point random point crossover.

Mutation is operated by a randomly selected one of the chromosomes bit and flipped. Probability of mutation, unlike crossover, is usually selected to be very small. Fig. 1.3 shows a simple mutation technique by flipping.

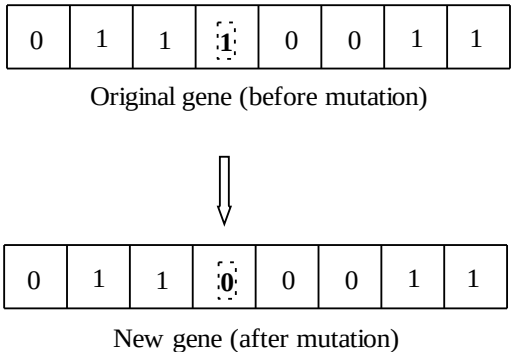


Figure 1.3: Genetic Algorithm mutation technique by selecting a bit randomly.

GA utilizes parallelism and has the ability to handle many complex optimization problems where the objective fitness function is either; linear, continuous, stationary, non-linear, discontinuous, non-stationary, etc. [9]. GA has some disadvantages as well in selecting the right parameters for the algorithms like rate of crossover and mutation, fitness function population, etc. It needs careful selection of these parameters for the algorithm to converge [6].

1.2.2.2. Simulated Annealing

SA is inspired by the annealing process of metal [16]. SA concerns the situation happening in metal when heated to a very high temperature and then allow the temperature to drop. This process usually causes the physical properties of the metal to change due to the change in the structure of the metal internally. The metal after gets cooled attains its new structure [17]. SA algorithm, for solving optimization problem was first introduced by Kirkpatrick and colleagues in 1983 to solve optimization problems. The main advantage of this algorithm is to overcome the shortcomings of some of the search methods, like gradient based, of converging to the local optima [6]. SA algorithm usually converges to global minima if sufficient randomness is utilized with a good cooling schedule. In SA algorithm, the idea of Markov chain is adopted by accepting any change that causes improvement in the objective function and some do not [17]. In an optimization problem, dealing with minimizing, any adjustment that reduces the objective function is accepted and those cause increase are also accepted based on the probability derived from Eq. (1.3).

$$p = e^{-\frac{\Delta E}{k_B T}}, \tag{1.3}$$

where ΔE is the change in energy, k_B is the Boltzmann's constant and for simplicity is set to 1, and T is the temperature for controlling the annealing process. In order to relate ΔE with Δf which is the change in objective function, Eq. (1.4) is used.

$$\Delta E = \gamma \Delta f, \tag{1.4}$$

where γ is a real constant and is set to be 1. Therefore, Eq. (1.3) becomes:

$$p = e^{-\frac{\Delta f}{T}}. \tag{1.5}$$

This is the probability for accepting any adjustment that causes change in the objective function and is also called transition probability [6]. In summary, SA can be described below [17]:

1. Stating the initial temperature: From Eq. (1.5), we can easily analyze the following:
   - as $T \to \infty$ then $P \to 1$.
   - as $T \to 0$ then $P \to 0$.

   So, at higher temperature, the probability of accepting any change in the objective function is very high, which means any worth solution is accepted but as the temperature approaches zero, so as the probability also, and under this condition rarely worth solutions will be accepted. Care should be taken in stating the initial temperature in order not to make the algorithm very slow or converge to the local optima [6].
2. Defining final temperature: The final temperature is stated to be very low $(10^{-10} \sim 10^{-5})$, so that no any worth change can be accepted [6].
3. Generate initial random solution.
4. Then looping continues until the stopping requirement is satisfied. It usually holds when the system gets cooled, that is when the final temperature is reached or when a very good fitness is obtained.
5. Then a neighbor solution is obtained from the current solution.
6. Decision is made on the new solution, if it could be accepted or not.
7. The temperature of the system is then reduced.
8. Examining the condition for the system to converge, if not repeat from step 4
9. Stop.

1.2.2.3. Artificial Bee Colony Algorithm

Exactly thirteen years after ACO algorithm was proposed by M. Dorigo, another important stochastic, population-based evolutionary algorithm was proposed in 2005 by Karaboga, called ABC algorithm. This algorithm is stimulated by the honey bee foraging behavior [18]. ABC algorithm is simple, robust and very flexible. In ABC, three components are important and they include: food source, employed and unemployed forager [19].

1. Food source: Each of the forager bees will choose a food source based on the properties it examines on the food source. These properties include; proximity to the hive, the amount of energy it contains, nectar taste, and simplicity of exploiting the energy. All these factors are represented by a single magnitude called quality or fitness.
2. Employed forager: These bees are sent to the food sources which are already explored to exploit. They communicate with the remaining bees in the hive about the distance, profitability and direction of the food source.
3. Unemployed forager: The bees staying in the hive are called unemployed bees and are divided into onlookers and scouts. A scout bee searches food source randomly in the surroundings. Onlooker bee's exploration is guided by the information received from the employed bee.

In ABC algorithms, the food sources are initialized randomly using Eq. (1.6) below.

$$x_i = x_j^{\min} + rand(0,1)(x_j^{\max} - x_j^{\min}), \tag{1.6}$$

where $i = 1,.....,NF$ and $j = 1,.....,D.$ NF is the number of food source and D is the number of parameter to be optimized, $x^{\min}$ and $x^{\max}$ represent the minimum and maximum range of the search space, respectively [18]. Each of the employed bees is directed to one source and carries the information about the quality of that source to the remaining bees in the hive. Each of the employed bees goes back to the neighborhood of its current source and checks for another source using Eq. (1.7) and evaluates its nectar [20].

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), \tag{1.7}$$

where j stands for a random integer ranges between 1 to D, i.e. $j = 1,...,D$ and $k \neq j$ always and is a randomly chosen index, $\phi_{ij} = rand(-1,1)$. However, after finding the second solution, a greedy selection is now applied between the two solutions in order to have a better

solution for the following generation. These better positions for each employed bee are selected by the onlooker in the hive based on their fitness values. The probability of selecting each position is stated in Eq. (1.8) below [20]:

$$p_i = \frac{fit_i}{\sum_{i=1}^{NF} fit_i}, \tag{1.8}$$

where fit_i is the fitness value of the food source which assigns the quality of the solution.

In general ABC stages can be summarized below:

1. State the values of control parameters which include; limit of the scout and the number of employed bees which is always equal to number of food sources.
2. Initialization: Here the position of the food sources are initialized which are the probable solutions of the optimization problem.
3. The employed bees are positioned on the food sources initialized and they evaluate their nectar.
4. The onlooker bees are positioned on the food source initialized and they evaluate their nectar.
5. The scout bees are then directed to randomly search for food sources.
6. Memorize the best food sources.
7. Stop.

1.2.2.4. Ant Colony Algorithm

In the year 1992, the ideas of ACO were put forward by the Italian researcher, Marco Dorigo and his colleagues as part of partial fulfillment for the award of his PhD thesis. Since then many difficult optimization problems are successfully being tackled by the use of this multi-agent ant-based approach [21]. ACO algorithms as the name implies has been stimulated by the actions of real living ants, especially by their foraging behavior, particularly their ability to determine from many the shortest path between their food sources and nest [22].

As the same time as the ants travel from their food sources to the nest and from the nest back to their food sources, they form on their way a trail by putting down on the ground a chemical substance called pheromone. Ants have the ability to sense pheromone and in probability choose the way to follow with the higher concentration of pheromone. The pheromone trail laid can guide the other ants for locating the food sources found by their mates and they

mostly stop searching at random, but instead going along the trail, strengthening the pheromone trail while going back if subsequently they find source of food [21]. Fig. 1.4 shows a typical example of how ants finds shortest route from their nest to food source. Ants coming from their food source separate at decision point A , randomly choose the upper path and some choose the lower [23]. The ants on the lower path, which is shorter, reach the food source and return to the colony before the ants that follow the upper and hence the pheromone on their way becomes more accumulated. The following ants from the colony at junction A will sense the pheromone from both ways and tend to follow the way that has more pheromone which is the lower path. Gradually, majority of the ants will be following the lower path since it is the shorter path.

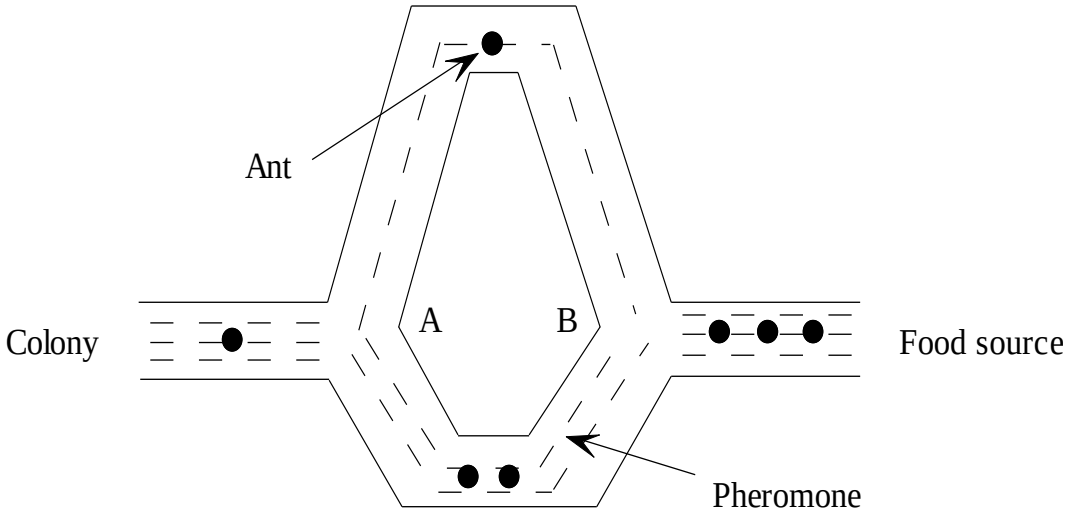


Figure 1.4: Ants find a shortest route.

ACO will be discussed in detail in the next chapter

CHAPTER TWO

PROBLEM STATEMENT

2.1. Introduction

With the advent of evolutionary algorithms as alternative approaches for solving many optimization problems, great achievements have been recorded to the solutions of many non-linear programming (NP) hard problems [24]. These algorithms include, but not limited to, the algorithms we explained before. These algorithms are being applied for solving 1-D optimization problems like: Travelling Salesman Problem (TSP) [21], quadratic assignment problem (QAP), sequential ordering problem (SOP) [21], etc. as well as 2-D problems that include Image Edge detection, Image clustering, etc.

In this thesis work, the performance of the four algorithms discussed in Chapter one which comprises GA, SA, ABC and ACO Algorithms will be implemented in TSP which is a 1-D problem and ACO will be applied for Edge Detection problem which is a 2-D problem.

TSP is a 1-D optimization problem that belongs to the class of Non-Linear Programming (NP-hard) problem of a salesman who wants to find the shortest closed tour of visiting all the cities exactly once and returning back to the starting point. TSP has many applications that include planning, logistics and manufacture of microchips. It can also be used in many areas such as; DNA sequencing as a sub-problem [21]. TSP is widely regarded problem according to the literature and has been considered by a lot of research works. TSP is selected to test the performance of these algorithms because of the following reasons:

1. It is being considered by researchers as one of the most important NP-hard problems that happen to be in various areas of application.
2. It is a problem in which the algorithms are not confused by many technicalities.
3. TSP is regarded by researches to be a standard laboratory for testing new algorithms. The performance of any new algorithms in TSP is used to measure its usefulness.

On the other hand, edge detection problem is a 2-D problem of finding areas where there are sudden changes in intensity which can be used in illustrating border lines in images. This important property, therefore, enables edge detection to be very important in many areas

including object or pattern recognition, image analysis, computer vision, and image processing [25]. ACO among the four algorithms, applied for 1-D case, is selected to be applied for 2-D case and the reason for that because it provides the best performance among the others in the 1-D case.

The remaining part of this chapter is devoted to the explanation of the two optimization problems to be considered for our work which are TSP and Edge Detection.

2.2. Travelling Salesman Problem

TSP is a 1-D NP-hard problem of finding the shortest closed tour of a visit for a set of cities given, and each city is visited exactly once. This problem is consort with distances between cities, or sometimes called costs. The salesman is expected to build a closed tour with minimum cost. The closed tour signifies a round-trip where by the salesman starts from a particular city and finally returns to that starting city after visiting all the cities and each city is visited once [26]. TSP is one of the discrete combinatorial optimization problems and can be represented using a weighted graph that connects all the cities, given as $G = (V, A)$, where V stands for the set of n cities, i.e. $V = \{v_1, \ldots, v_n\}$, and A is the set of all paths (edges) or arcs that connects all cities. The set, A can be donated as $A = \{(s,t) : s,t \in V\}$ and each of the edges $(s,t) \in A$ is given a distance $d_{s,t}$ as the distance between two cities s and t. If in any given TSP and in at least one of the arcs $d_{s,t} \neq d_{t,s}$, the problem is asymmetric TSP. Else if for all arcs $d_{s,t} = d_{t,s}$ the problem is called symmetrical TSP. Generally, TSP is aimed to compute the permutation π of the node indices $\{1, 2, 3, \ldots, n\}$ which minimizes the equation given in Eq. (2.1) below:

$$f(\pi) = \sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(1)}. \tag{2.1}$$

TSP has been solved using numerous methods which are usually of two categories; approximation and exact algorithms [27]. Approximation methods use heuristics and iterative improvement processes, unlike exact algorithms that use mathematical models. Exact algorithms attempt to find a good solution of an optimization problem and attest their optimality when tested for small instances. Approximation algorithms have the advantage of producing satisfactorily good solutions within short time [28]. Approximation algorithms can be divided into constructive heuristics and improvement heuristics. Solving TSP using

Insertion heuristic, Greedy heuristic, nearest neighbor, etc. are all constructive heuristic methods, while $k-\text{opt}$, EAs, and metaheuristic based algorithms like SA, ABC, ACO, etc. are all belong to improvement heuristics. On the other hand, exact methods based instances include: Lagrangian Relaxation, Integer Linear Programming, etc. [26].

2.2.1. Genetic Algorithm for Travelling Salesman Problem

GA has been applied for solving many optimization problems. TSP is one of the common problems where such algorithm has been applied [29]. The way we applied it for TSP is summarized below:

Populations of individuals' solutions are initially created randomly which are regarded as our tours and their codification is called chromosome. This chromosome is an arrangement of symbols that represent cities. These tours are our initial solutions and each one signifies a closed path of visiting all the cities. Two of these tours which have the minimum distance (highest fitness) were selected to be the parent-tours and crosses together to produce another two solutions called children-tours, which expectantly will be better that the two parents solutions. The selection criteria used is similar to Eq. (1.2). The crossover used in our analysis is a single point crossover explained in Section 1.2.2.1 of Chapter One because it is simple and gives good results. In order to avoid generating a solution that is identical to any of the parents, the new solutions produced were mutated. The mutation used is swap mutation; where by the sequence of the two cities codified in the chromosome were interchanged. The two longer tours in the population were substituted by the two children tours produced. The process is repeated by continually generating new children tours until termination criterion is met.

Even though, this algorithm shows ability to solve TSP but the results obtained (as shown in Chapter 4) are not satisfactory and hence another algorithm called SA, which is inspired by the annealing process of metal, was tested to solve the same problem. The implementation process is detailed below.

2.2.2. Simulated Annealing for Travelling Salesman Problem

SA is also another metaheuristic algorithm which, if implemented properly and took care of all the parameters selection, can ensure near optimal solution of the TSP. Below, the summary of procedure we used in applying SA to solution of TSP is given.

The initial temperature of the system was stated to be the same as the maximum cost (worth cost) and allowed to drop quickly until 50% of the worth solutions were accepted, and then the temperature was used as our initial temperature. The solution was represented as a series of nodes and these nodes are arranged in the order they were visited. Each node is written once in the list. In this solution representation node 1 always comes the first but is usually not stated in the list but is assumed to be. The last node is also1. For example if the solution is represented by the order: "$4, 5, 3, 6, 2, 1$" it means the tour is started from city 1 to city $4, 5, 3, 6, 2$ and lastly go back to city1. A feasible solution to the TSP was found using greedy method; it begins the tour with the first node and to the most nearest node until the tour is completed. At each node find the closest node to it and not visited before, until a closed tour is formed. This solution obtained is the initial solution. From this solution, a neighbor solution is produced by randomly interchanging two pairs of nodes. Node 1 is not among the nodes to be interchange always. For example, if position 3 and 2 are randomly selected to swap, then the new tour will be "$4, 5, 2, 6, 3, 1$". This neighbor solution, which is the new route produced ("$4, 5, 2, 6, 3, 1$"), is compared with the current route ("$4, 5, 3, 6, 2, 1$"). If the length of the new route is shorter than the current route, the new route becomes the current route and is checked whether it is shorter than the previous routes found so far. The new route is saved as the best route if it is shorter than all the routes found so far. If the length of the new route is higher than the old route, the difference in length between the two tours is used to calculate the transition probability explained in (1.5) for accepting or rejecting the new route. A random number generated between '0' and '1' is compared with this transition probability. If the probability value is greater than the random number, the change is accepted and the new route becomes the current one, otherwise, it is rejected and the old route is maintained. The temperature of the system is then updated using $T \leftarrow alpha * T$ where $alpha$ is the geometric annealing factor. The algorithm continues until the temperature approaches zero.

The results obtained using this algorithm show significant improvement when compared with those of the GA, but since ABC algorithm was also successfully implemented for various instances in solving TSP problem and was found to be robust in many optimization problems, it is also introduced to solve the same data sets of the problems.

2.2.3. Artificial Bee Colony for Travelling Salesman Problem

ABC Algorithm is inspired by the foraging behavior of honey bees and has been applied to solve many optimization problems. Here, we introduced the way we used in solving the TSP [26].

A number of bees, $(N\ bee)$, which is equal to the same number of cities, is initialized. Afterwards bees are allowed to build a closed tour randomly. A cycle is made after all the bees have completed the tour. In the hive each bee, before leaving, randomly selects one of the paths explored by another bee based on the dances watched. This path called "$Preferred-path$" denoted as ω, will lead the bee in forming its tour. At city i, a bee will have two sets of paths to go, which are; "$allowed-city$" denoted as $A_i(t)$, and "$preferred-city$" denoted as $F_i(t)$. $A_i(t)$ contains all the cities that can be visited from i including that of $F_i(t)$, and $F_i(t)$ is a set of a single city which is suggested to be visited by ω from i. Let $\omega(n)$ represents the $n-th$ element in ω. From the hive at the start of the tour $F_H(0)=\{\omega(1)\}$ and if at a given time t, $\omega(n)$ is the recently visited city then $F_{\omega(n)}(t)=\{\omega(n+1)\}$.

Throughout the foraging activity, a bee goes from one city to another up to the end. Here, the transition from city i to j is guided by transition rule; which is based on the arc fitness and heuristic distance. At a given time, the arc fitness of all the paths to cities, that can be visited from that specific city, is evaluated and the arc which belongs to the preferred $Preferred-path$ is given the highest value. This causes the bee to select the city which is part of the $Preferred-path$ to visit next. However, for the heuristic distance effect, a bee always chooses the closest city to its current position to visit. The transition rule for moving from city i to j is defined as:

$$p_{ij}^{bee}(t)=\begin{cases}\dfrac{\zeta_{ij}^{\alpha}(t).\eta_{ij}^{\beta}(t)}{\sum_{l\in A_i(t)}\zeta_{il}^{\alpha}(t).\eta_{ij}^{\beta}(t)}, & \text{if } \ j\in A_i(t)\\ 0, & \text{otherwise}\end{cases} \tag{2.2}$$

where $\zeta_{ij}(t)$ is the arc-fitness from city i to j, α and β are parameters that control the influence of arc-fitness and heuristic distance respectively. $\eta_{ij}=\dfrac{1}{d_{ij}}$ is the heuristic distance.

The arc-fitness is given by:

$$\zeta_{ij}(t)=\begin{cases}\sigma, & j\in F_i(t)\\ \dfrac{1-\sigma\left|A_i(t)\cap F_i(t)\right|}{\left|A_i(t)\right|-\left|A_i(t)\cap F_i(t)\right|}, & j\notin F_i(t)\end{cases}\quad \forall j\in A_i(t),\ 0\le\sigma\le 1 \tag{2.3}$$

From this equation $\left|A_i(t)\cap F_i(t)\right|$ is set to '1' when the case is common in both $A_i(t)$ and $F_i(t)$, or else set to '0'. The parameter σ stands for the probability of selecting a city in the $Preferred-path$.

Supposing a bee has five cities to visit, as $V=\{1,2,3,4,5\}$. It starts from hive, H and is provided with a $Preferred-path$ as, $\omega=$"2,1,3,4,5,2". This path is witnessed from one of the bees via waggle dance in the hive. In order to push the bee to move to the first city in ω, the distance between the hive and any other city is assumed the same. In this example, heuristic distance has no effect. At $t=0$ a bee leaves its hive and go the first city to be visited and from $\omega=$"2,1,3,4,5,2", two city sets can be obtained. These sets are: $A_H(0)=\{1,2,3,4,5\}$ and $F_H(0)=\{2\}$. The fitness values are allocated on all the arcs connecting the hive and cities in $A_H(0)$. σ is given to arc $(H,2)$ and $\dfrac{(1-\sigma)}{4}$ is given to: $(H,1),(H,3),(H,4)$ and $(H,5)$. Usually, the fitness values at a given time t from city i can be written in matrix, $\Phi_i(t)$ having a size of $1\times\left|A_i(t)\right|$ and entries, symbolized as ζ_{ij}, represent the arc fitness from city i to j. Below is a matrix for the arc fitness from hive to city j at $t=0$ and $j\in A_H(0)$. $\Phi_H(0)=[\zeta_{H1}\ \zeta_{H2}\ \zeta_{H3}\ \zeta_{H4}\ \zeta_{H5}]$ or $\left[\dfrac{1-\sigma}{4}\ \ \sigma\ \ \dfrac{1-\sigma}{4}\ \ \dfrac{1-\sigma}{4}\ \ \dfrac{1-\sigma}{4}\right]$. The sum of all the arc fitness values associated to the paths connecting i to the other cities is '1' as shown below:

$$\sum_{j\in A_i(t)}\zeta_{ij}(t)=1. \tag{2.4}$$

If city 2 is previously selected to be visited at $t=1$, $A_2(t)=\{1,3,4,5\}$ and $F_2(1)=\{1\}$, the arc fitness values matrix will be:

$\Phi_2(1)=\left[\zeta\ \ \dfrac{1-\zeta}{3}\ \ \dfrac{1-\zeta}{3}\ \ \dfrac{1-\zeta}{3}\right]$. This idea always directs the bee towards following the path which is part of the preferred by assigning the highest fitness value which is ζ and all the rest as $(1-\zeta)$.

Suppose 5 is selected at $t=1$, then at $t=2$ the sets of allowed cities will be $A_5(2)=\{1,3,4\}$ and $F_5(3)=\{2\}$. Under this condition all the arcs connecting to 5 with $A_5(2)$ cities will have equal values of fitnesses as shown below:

$$\Phi_5(2)=\left[\zeta_{51}\ \ \zeta_{53}\ \ \zeta_{54}\right]=\left[\begin{array}{ccc}\frac{1}{3} & \frac{1}{3} & \frac{1}{3}\end{array}\right].$$

At the hive, a bee dances if and only if it finds a shorter path compared to the earlier tours it made. So, not all the bees dance in the hive when returned. The period taken by a bee i to dance in the hive, when a tour better than its previous tours is found, is given as

$$D_i = K_S.\frac{pf_i}{pf_{col.}}, \tag{2.5}$$

where $pf_i=\frac{1}{L_i}$ is the profitability of a bee i, K_S is the scaling factor of the dance and $pf_{col.}=\frac{1}{n}\sum_{i=1}^{n} pf_i=\frac{1}{n}\sum_{i=1}^{n}\frac{1}{L_i}$, is the average profitability of the bees that perform waggle dances in the hive and n stands for the number of bees that dance.

However, before any bee selects a tour to follow, it decides either to follow or not with the probability of follow, $p_{fol.}$, given as:

Table 2.1: Probability of follow.

Profitability score	$p_{fol.}$
$pf_i < 0.5\, f_{col.}$	0.60
$0.5\,pf_{col.} \le pf_i < 0.65\,pf_{col.}$	0.20
$0.65\,pf_{col.} \le pf_i < 0.85\,pf_{col.}$	0.02
$0.85\,pf_{col.} \le pf_i$	0.00

The probability of follow from the Table 2.1 is changing with the change in pf_i and $pf_{col.}$. In a situation where by the $p_{fol.}=0.00$ a bee will trace back its previous path. It will not select any new path.

This algorithm presents better results when compared with the results obtained by GA and SA. However, ACO is another optimization algorithm which can easily be applied to solving TSP.

2.2.4. Ant Colony Optimization for Travelling Salesman Problem

ACO algorithm is inspired by the foraging behavior of ants, and it has been applied for solving many optimization problems. Here we applied it to solve TSP as follows:

Initially, all ants were placed on the cities randomly and the initial values of the pheromone trail $\tau^{(0)}$ are assigned on each city. Each ant will use the transition formula given below to move from one city to another until it returns to the starting city (node). Ant k uses Eq. (2.6) to define the probability of moving from city i to j.

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{\tau_{ij}^{\alpha}(t).\eta_{ij}^{\beta}(t)}{\sum_{l\in N_i^k}\tau_{il}^{\alpha}(t).\eta_{ij}^{\beta}(t)}, & \text{if } \; j\in N_i^k \\ 0, & \text{otherwise}\end{cases}, \tag{2.6}$$

where N_i^k is the set of all cities that can be visited from i excluding the cities passed. It can be simply written as $N_i^k=\left\{N-tabu^k\right\}$, α and β are constant values used to control the influence of pheromone and visibility, respectively, and $\eta_{ij}=\dfrac{v}{d_{ij}}$ is the visibility, where d_{ij} is the distance between cities i, j and v is a constant.

The next step, after all the ants have finished building the tours, all the paths are globally updated using:

$$\tau_{ij}(t+n)=\rho\tau_{ij}(t)+\sum_{k=1}^{n}\Delta\tau_{ij}^{k}, \tag{2.7}$$

where ρ is the evaporation coefficient and τ is a constant value $(0\sim1)$,

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k} & \text{if } ij \text{ belongs to the tour made by} k \\ 0, & \text{otherwise}\end{cases},$$

where L_k is the total tour length performed by ant k and Q is a constant value associated with the amount of trails deposited.

The length of the tours obtained using this algorithm happens to be shorter than those of the other three algorithms. It gives minimum costs than GA, SA and ABC and, hence, we will use the ACO algorithm in detecting the edges of image since it is an optimization problem.

2.3. Edge Detection

Edge detection is considered as an important tool in image processing. It has been used an important pre-processing tool in feature extraction and object segmentation [30]. Edges are regions where there are sudden changes in intensity and are used to define boundaries in an image. Through the edge detection process, useful information in an image are reserved and useless ones are eliminated [31]. Researchers have succeeded in introducing a lot of edge detection methods and each one is purposed to fit a distinct types of edges. Some of these methods are; Canny, Laplacian of Gaussian, Prewitt operator, Robert operator, Sobel operator [32]. Up to our knowledge, Canny always provides the best results. Usually, these operators are matrix-based and they are used to perform gradient operation on image area in order to find the intensity of variance between pixels [30]. When dealing with these operators for edge detection usually four steps are followed;

1. Smoothing: This is the process of removing the available noise in the image. Care should be taken in smoothing an image in order not to damage some edges.
2. Enhancement: Enhancement or sharpening is applied after smoothing to improve the quality of the image's edges. A filter can be used for this purpose.
3. Localization: Two important tools are usually needed in this steps which are; thinning and linking. Localization involves finding the precise location of an edge.
4. Detection: This involves using process like intensity threshold to finally obtain the edge pixels.

Recently a lot of research work has shown the ability of evolutionary algorithms, such as ACO and ABC, to be effectively used for tackling edge detection problems. Examples of these works include, the work in [18] for edge detection of CNN based imaging sensors using ABC for designing a novel cloning template. Also, the work in [25] for solving edge detection problems, especially, with noisy images using ACO and DWT. The results obtained using these algorithms show significant improvement.

2.3.1. Ant Colony Optimization for Edge Detection

One of several techniques that are successfully being applied for image edge detection problem is the ACO algorithms [33]. This optimization technique is based on the foraging behavior of living ants [6]. In this technique, ants deposit a chemical substance, called pheromone, on the ground which is used for communication among them. Many optimization problems have been solved using ACO [26]. ACO is applied in edge detection problem to obtain the edge information; which is very vital in understanding the information stored in the image [34]. Such method concerns the ants movements driven by the local variation of image pixels to store positions where there is a change in the intensity value in their memory and update the pheromone matrix. It starts with initialization state where the pheromone matrix is initialized and the heuristic matrix is calculated. The other stages are: construction, updating and decision. These stages are given in detailed below:

Table 2.2: ACO algorithm for Edge Detection.

---

1. Initialization phase

Define the initial pheromone matrix, calculate the heuristic matrix.

2. Construction phase

for $n = 1:N$ place all the ants on the image randomly.
for every movement $l = 1:L$
for every ant $k = 1:K$
move to the next pixel in its neighborhood and update pheromone locally.
end
end
update visited pixels
end

3. Decision phase

binary decision is finally made on the updated pheromone matrix on each entry weather is an edge or not to obtain the image edge detected results.

2.3.1.1. Initialization stage

In the initialization stage, the artificial ants are randomly dispatched on the image. A pheromone matrix whose size is the same as that of the input image is formed. The entries of this matrix are set to very small initial values and denoted as $\tau^{(0)}$. Another matrix called heuristic matrix is also formed, the entries of this matrix are calculated based on the image intensity values of the pixels which rely on clique. The heuristic information at pixel location (i, j) is defined as:

$$\eta_{i,j} = \frac{V_c(I_{i,j})}{Z}, \tag{2.8}$$

where $V_c(I_{i,j})$ is the variation in intensity value between the pixel $I_{i,j}$ and the group of local neighboring pixels and computed as shown in Fig. 2.1, Z is the normalization coefficient and is given as, $Z = \sum_{i=m^{th}row} \sum_{j=n^{th}col} V_c(I_{i,j})$.

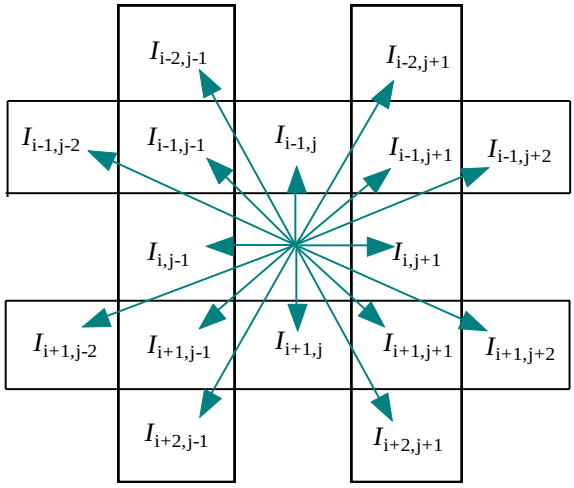


Figure 2.1: Neighbors of pixels $I_{i,j}$.

The variation in intensity given in Eq. (2.8) is computed using these neighbor pixels as,

$$\begin{aligned} V_c(I_{i,j}) = f\Big(&\left|I_{i-2,j-1} - I_{i+2,j+1}\right| + \left|I_{i-2,j+1} - I_{i+2,j-1}\right| + \left|I_{i-1,j-2} - I_{i+1,j+2}\right| + \left|I_{i-1,j-1} - I_{i+1,j+1}\right| + \\ &\left|I_{i-1,j} - I_{i+1,j}\right| + \left|I_{i-1,j+1} - I_{i+1,j-1}\right| + \left|I_{i-1,j+2} - I_{i+1,j-2}\right| + \left|I_{i,j-1} - I_{i,j+1}\right|\Big). \end{aligned} \tag{2.9}$$

This variation is higher when the pixels are located at edges. However, the image borders do not have complete neighbor pixels, for example pixel positioned at $(1,1)$ does not possess pixels located at the north and west of it and, therefore, the clique is not complete. These border pixels are assigned a very small value so that the area will be less attractive for ants. Equations (2.10) – (2.13) are carefully taken into consideration in finding the function, $f(.)$, stated in Eq. (2.9).

$$f(x) = \lambda x \quad \text{for } x \geq 0, \tag{2.10}$$

$$f(x) = \lambda x^2 \quad \text{for } x \geq 0, \tag{2.11}$$

$$f(x) = \begin{cases} \sin\left(\dfrac{\pi x}{2\lambda}\right) & \text{for } 0 \leq x \leq \lambda \\ 0 & \text{else} \end{cases}, \tag{2.12}$$

$$f(x) = \begin{cases} \dfrac{\pi x \sin\left(\dfrac{\pi x}{\lambda}\right)}{\lambda} & \text{for } 0 \leq x \leq \lambda. \\ 0 & \text{else} \end{cases} \tag{2.13}$$

Where λ is a parameter that is used to change the shape of these functions [34].

2.3.1.2. Construction stage

During this stage, at every n^{th} construction step, an ant is randomly chosen from the total ant number, K. This ant moves from one point to another on the image for L steps. The transition of ant from pixel location (l,m) to (i,j) is based on the equation below:

$$p_{(l,m),(i,j)}^{(n)} = \frac{(\tau_{i,j}^{(n-1)})^{\alpha}(\eta_{i,j})^{\beta}}{\sum_{(i,j)\in\Omega_{(l,m)}}(\tau_{i,j}^{(n-1)})^{\alpha}(\eta_{i,j})^{\beta}}, \tag{2.14}$$

where $\tau_{(i,j)}^{(n-1)}$ represents the quantity of pheromone trail at pixel (i,j), $\eta_{i,j}$ is the heuristic information at pixel (i,j), α and β are parameters that stand for the influence of pheromone trail and heuristic information, respectively. $\Omega_{(l,m)}$ represents all the pixels that can be visited from pixel (l,m), i.e. its neighborhood pixels.

2.3.1.3. Update stage

During this stage, two types of pheromone update are usually carried out; which are local update and global update. Local update is performed after each ant, k, moves during the construction step. This update is performed using Eq. (2.15),

$$\tau_{i,j}^{(n-1)} = \begin{cases} (1-\rho).\tau_{i,j}^{(n-1)} + \rho.\Delta_{i,j}^{(k)}, & \text{if } (i,j) \text{ is visited by } k^{th} \text{ ant} \\ \tau_{i,j}^{(n-1)}, & \text{otherwise} \end{cases}, \tag{2.15}$$

where ρ is the pheromone evaporation rate and $\Delta_{i,j}^{(k)}$ is determined from the heuristic information, i.e. $\Delta_{i,j}^{(k)} = \eta_{i,j}$. The global update is done after all the ants have finished the movement steps. The update is given in Eq. (2.16) as,

$$\tau_{i,j}^{(n)} = (1-\psi).\tau_{i,j}^{(n-1)} + \psi.\tau_{i,j}^{(0)}, \tag{2.16}$$

where ψ is the pheromone decay coefficient and $\tau_{i,j}^{(0)}$ is the initial pheromone matrix at pixel location (i, j).

2.3.1.4. Decision stage

In this stage, the updated pheromone matrix is used to obtain the detected edges by defining a threshold value, T_h. Binary decision is usually made on each pixel of the final pheromone matrix $\tau^{(N)}$ to verify if it is an edge or not. The threshold, T_h, is computed adaptively based on the method in [34]. This process is as follows:

Step 1: The initial threshold $T_h^{(0)}$ is computed at $iter = 0$ as,

$$T_h^{(0)} = \frac{\sum_{i=1}^{M_1}\sum_{j=1}^{M_2}\tau_{i,j}^{(N)}}{M_1 M_2}, \tag{2.17}$$

where M_1 and M_2 are the row and column sizes of the pheromone matrix, respectively, and they also denote the size of the original image.

Step 2: The entries of the pheromone matrix $\tau^{(N)}$ are then grouped into two; the first group contains all the entries below the threshold value $T_h^{(iter)}$, and the second group contains the rest of the entries. The mean value of these two groups of pixels is then computed as:

$$m_L^{(iter)} = \frac{\sum_{i=1:M_1} \sum_{j=1:M_2} g^L_{T_h^{(iter)}}(\tau_{i,j}^{(N)})}{\sum_{i=1:M_1} \sum_{j=1:M_2} h^L_{T_h^{(iter)}}(\tau_{i,j}^{(N)})}, \tag{2.18}$$

$$m_U^{(iter)} = \frac{\sum_{i=1:M_1} \sum_{j=1:M_2} g^U_{T_h^{(iter)}}(\tau_{i,j}^{(N)})}{\sum_{i=1:M_1} \sum_{j=1:M_2} h^U_{T_h^{(iter)}}(\tau_{i,j}^{(N)})}, \tag{2.19}$$

where

$$g^L_{T_h^{(iter)}}(x) = \begin{cases} x, & \text{if } x \le T_h^{(iter)} \\ 0, & \text{otherwise} \end{cases}, \qquad h^L_{T_h^{(iter)}}(x) = \begin{cases} 1, & \text{if } x \le T_h^{(iter)} \\ 0, & \text{otherwise} \end{cases},$$

$$g^U_{T_h^{(iter)}}(x) = \begin{cases} x & \text{if } x \ge T_h^{(iter)} \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad h^U_{T_h^{(iter)}}(x) = \begin{cases} 1, & \text{if } x \ge T_h^{(iter)} \\ 0, & \text{otherwise} \end{cases}.$$

Step 3: set iteration as $iter = iter + 1,$ and the threshold value T_h is updated using:

$$T_h^{(iter)} = \frac{m_L^{(iter)} + m_U^{(iter)}}{2} \tag{2.20}$$

Step 4: In this step, binary decision is made on the pixels. If $\left|T_h^{(iter)} - T_h^{(n-1)}\right| < e,$ terminate the algorithm and check each pixel if it is an edge or not. Else go back to step 2. The decision on pixel $E_{i,j}$ is made based on the standard below:

$$E_{i,j} = \begin{cases} 1, & \text{if } \tau_{i,j}^{(N)} \ge T^{(iter)} \\ 0, & \text{otherwise} \end{cases} \tag{2.21}$$

where e is the decision tolerance value and usually is selected to be a small positive number.

CHAPTER THREE

PROPOSED ALGORITHMS

3.1. Overview

Even though the ACO algorithm has shown significant performance in edge detection problem, this performance can be enhanced further by applying a DWT on the input image. This technique is presented in detail in the rest of this chapter.

3.2. The Proposed Algorithms

The edge detection method explained in Section 2.3.1 shows ability to detect edges of an image. However, some results obtained using this method do not have enough information to clarify the image concerned, especially, when the edge results are hampered by breaking up or fragmentation, or when the original image is contaminated with noise. To overcome these problems, we propose a new algorithm based on DWT and ACO. The DWT is applied to the image as the preprocessing stage before the conventional ACO algorithm is applied. The edge detection results obtained using this technique show substantial improvements, more especially, when the original image is noisy. Also, another approach is proposed to improve the edge detection results by detecting the edges of each of the four sub-images obtained by decomposing the original image using 2D-DWT, independently, and then, using the 2D-inverse DWT (2D-IDWT), the detected edges are combined. The two proposed algorithms are presented below.

3.2.1. Discrete Wavelet Transform-based ACO for Edge Detection

In an attempt to improve the edge detection results obtained by the conventional ACO, the concept of wavelet transform is introduced into the system. Wavelet transform is a process of decomposing, analyzing and displaying the processed major components of an image and leaving the rest of the components without processing [25]. Wavelet transform has been applied in various edge detection techniques [35]. In an effort to improve edge detection results, we propose applying DWT to the image as a preprocessing step before applying the conventional ACO algorithm in order to be capable of detecting the undetected or badly

detected edges, especially, when the image is contaminated with noise. Fig. 3.1 shows the flowchart of the proposed algorithm.

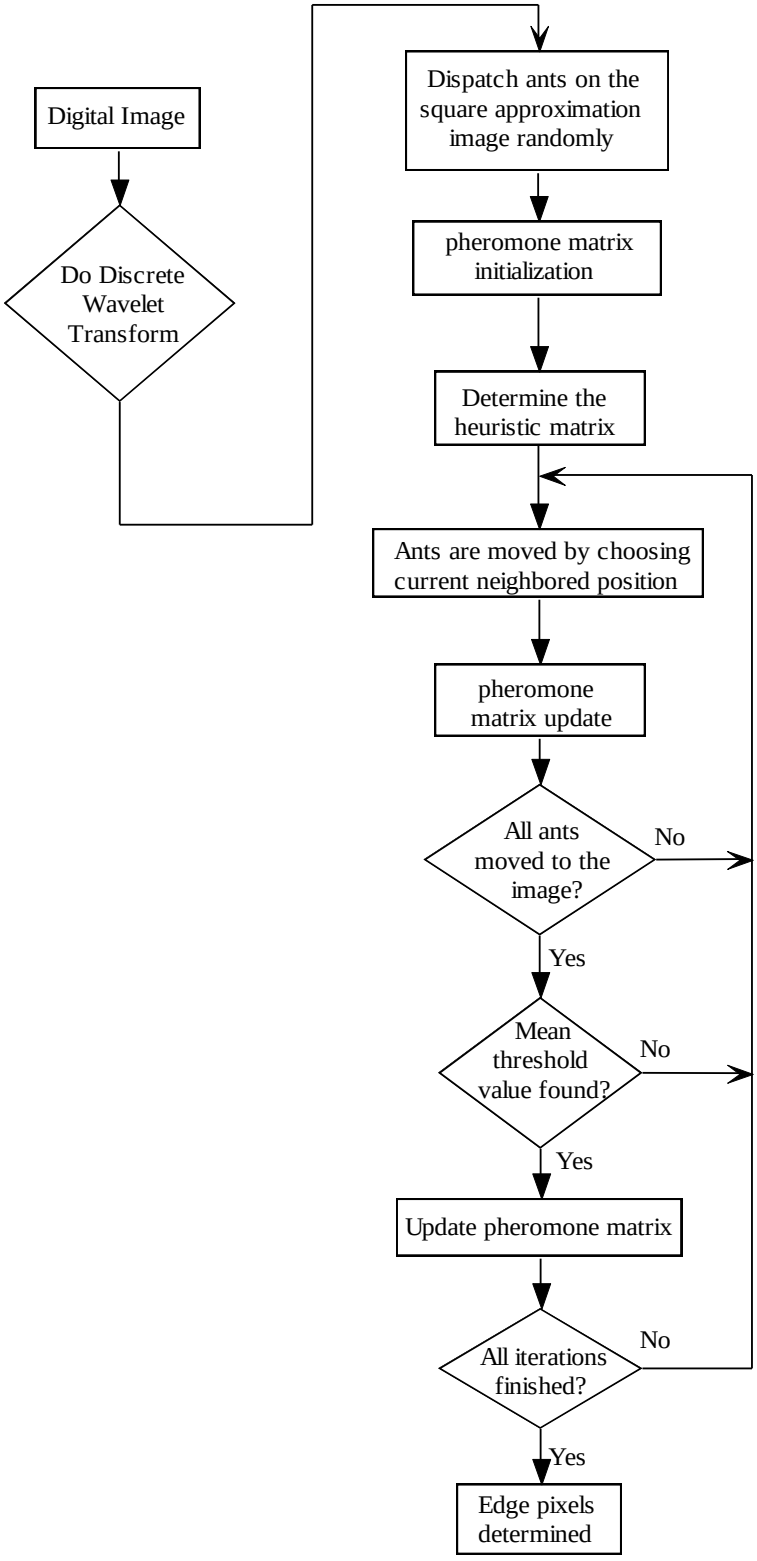


Figure 3.1: Flowchart of the proposed Discrete Wavelet Transform-based ACO for Edge Detection.

As shown in the flowchart above, in this proposed algorithm the output image from the DWT is the input image to the conventional edge detection techniques using ACO explained in Section 3.2.1. The performance of the proposed algorithm shows clear and significant improvement in edge detection results, especially, when the input image is corrupted by noise.

3.2.2. DWT Sub-Band Fusion using ACO for Edge Detection

Even though the proposed algorithm shows a good capability of detecting edges that may not be detected by the conventional ACO, this performance can be enhanced more by using the approximation coefficient and the three details coefficients submatrices, resulted from the DWT. Each of these four submatrices is processed separately, and the resulting four matrices are recombined using the IDWT. The details of this method are below.

3.2.2.1. Two Dimensional Discrete Wavelet Transform

The digital image can be decomposed at j level into four components using 2D-DWT. The components are; the approximation coefficient at level $j+1$ and the three details coefficient which includes: horizontal coefficients, vertical coefficients and diagonal coefficients. The 2D-DWT decomposition of an image is shown in Fig. 3.2, [36, 37]:

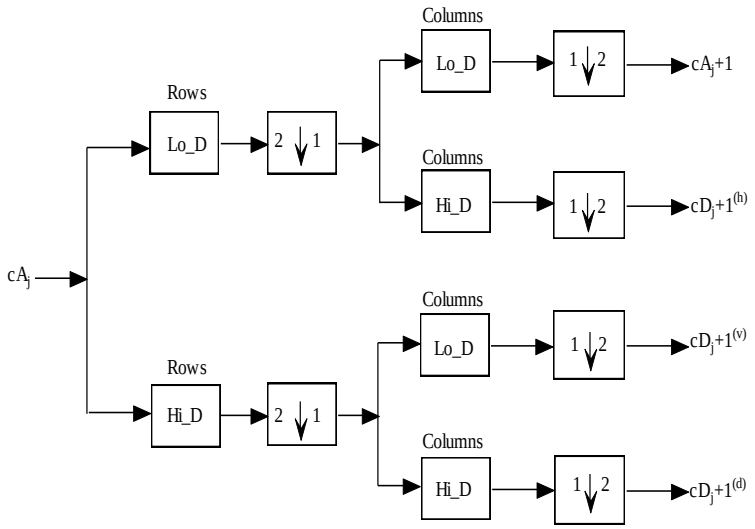


Figure 3.2: Two-dimensional discrete wavelet image decomposition.

where $2\downarrow 1$ down samples columns of image, cA_j and $1\downarrow 2$ down samples rows; Lo_D and Hi_D are the decomposition low pass filter and high pass filter for convolutions with rows and columns of cA_j. The components; $cA_{j+1}, cD^{(h)}_{j+1}$, $cD^{(v)}_{j+1}$ and $cD^{(d)}_{j+1}$ are always of the same size of the image and they represent the approximation coefficient, the horizontal coefficient, the vertical coefficient and the diagonal coefficient matrices, respectively. The approximations represent the high scale and components of the image with low frequency while the details represent the opposite; low scale components of the high frequency. This process involves 1-D convolutions of rows and columns of the image, cA_j with the decomposition low pass filter, Lo_D and decomposition high pass filter, Hi_D.

3.2.2.2. Two Dimensional Inverse Discrete Wavelet Transform

2D-IDWT can be used to reconstruct back the original image from high and low frequency components obtained from the decomposition process of 2D-DWT, [36].

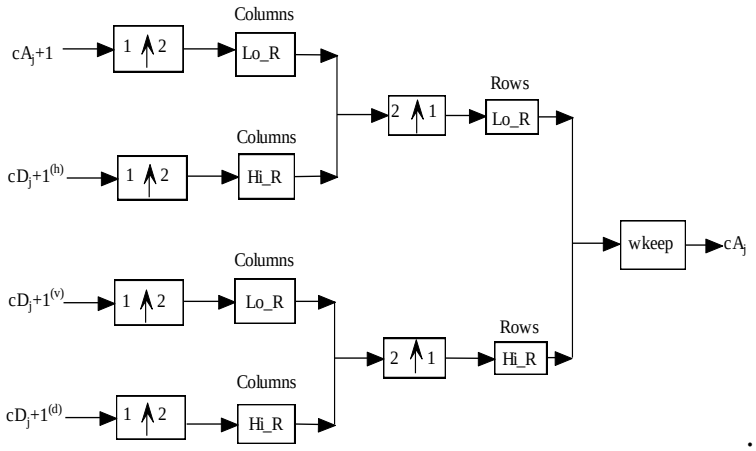


Figure 3.3: Two-dimensional discrete wavelet image reconstruction.

where $1\uparrow 2$ and $2\uparrow 1$ are for up sampling rows and columns respectively. The stage Lo_R signifies the reconstruction low pass filter and Hi_R stands for the reconstruction high pass filter.

These processes of decomposition and recombination of a digital image using 2D-DWT and 2D-IDWT, respectively, were used in our proposed algorithm. The digital image is decomposed in the first stage of the algorithm into four sub images and each of these sub images is applied to edge detection algorithm using ACO method explained in Section 3.2.1 above. The detected edge results of these four sub images are recombined together to produce

the final edge detection result using 2D-IDWT. The flowchart of the proposed algorithm is shown in Fig. 3.4.

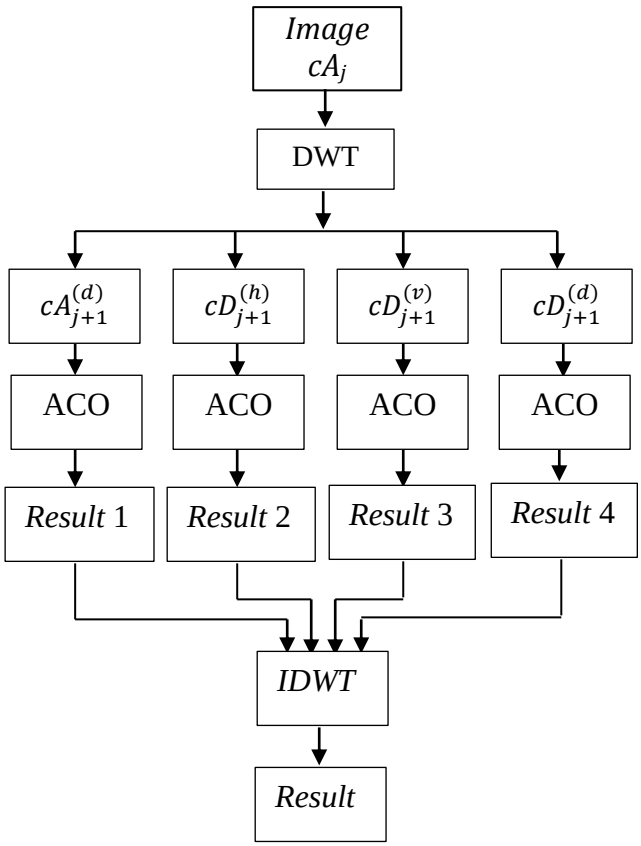


Figure 3.4: Flow chart of the proposed DWT Sub-Band Fusion using ACO for Edge Detection.

CHAPTER FOUR

SIMULATION RESULTS

4.1. Overview

This chapter presents the results that compare the performance of the GA, SA, ABC and ACO algorithms in TSP. It also presents the results that compare the performance of the proposed algorithm with that of the ACO algorithm in edge detection. MATLAB software (version 2012) was used for testing all the algorithms.

4.2. Travelling Salesman Problem

4.2.1. The Same Number of Cities with Independent Generations

In this experiment, three different data sets were randomly generated and each of them contains 30 cities. Each of these data sets was considered separately and the minimum distance of visiting all the 30 cities once was measured using GA, SA, ABC and ACO algorithms. All these algorithms were run for 200 independent runs for each of the data sets. The positions of the cities and implementation details for each of the algorithms are summarized below:

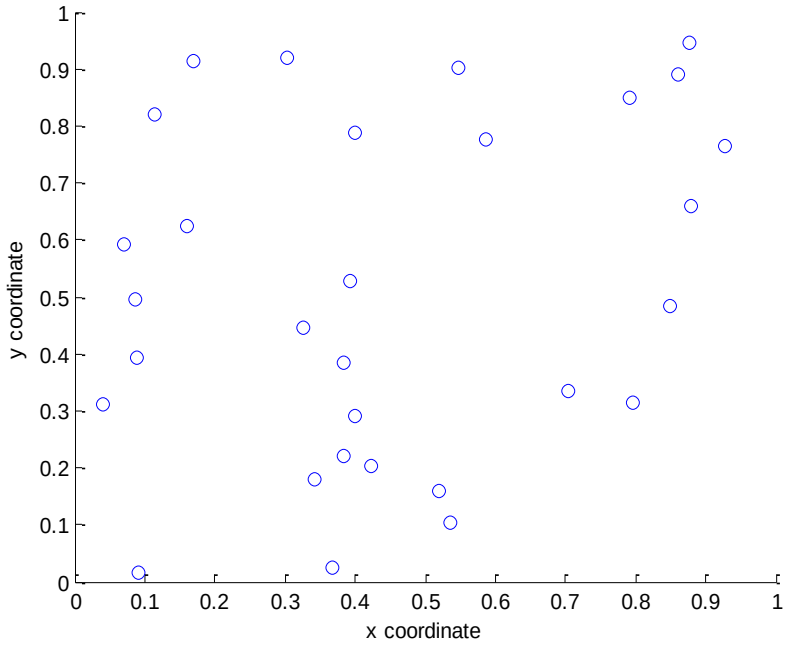


Figure 4.1: Random cities positions of data set 1.

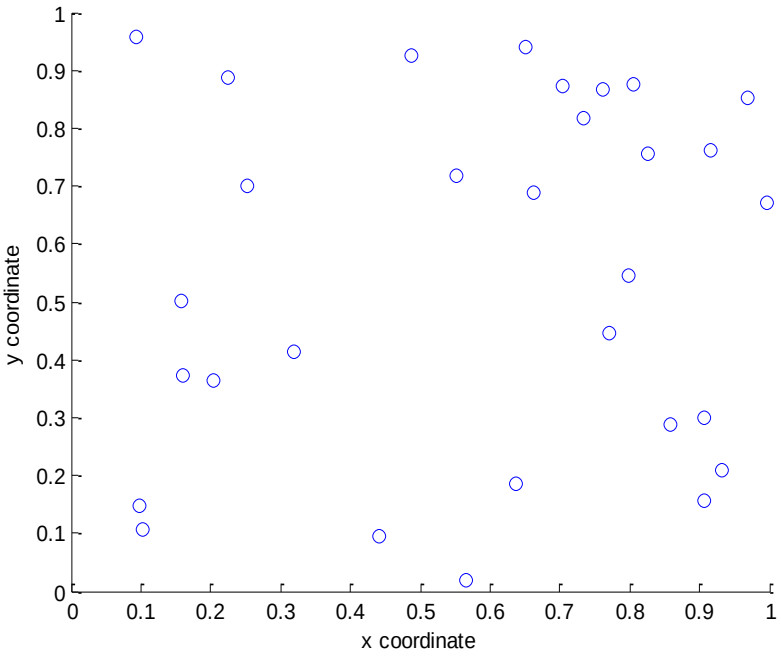


Figure 4.2: Random cities positions of data set 2.

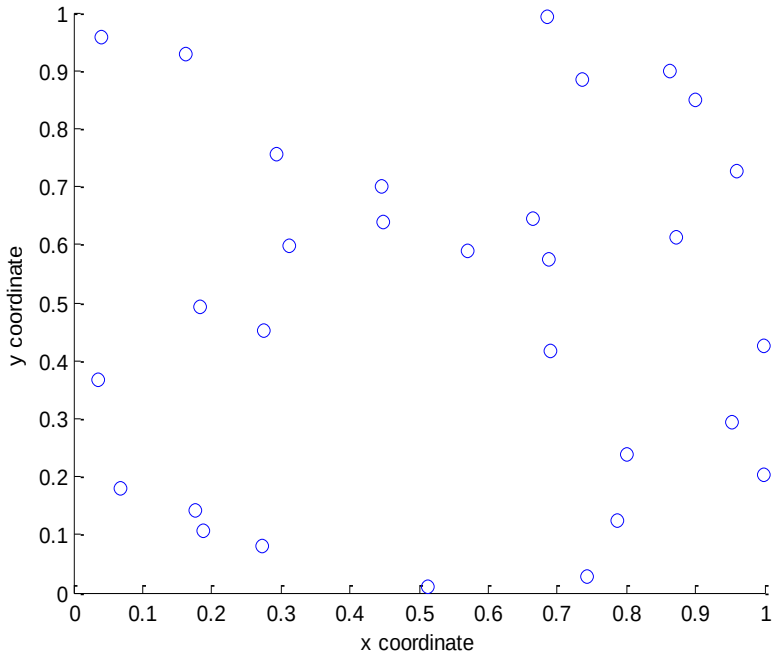


Figure 4.3: Random cities positions of data set 3.

All of the algorithms are implemented with the following parameters: Population size is set equal to the number of cities (30). For the GA: Swap mutation and single point crossover are used. For the SA algorithms: The geometric annealing factor $alpha = 0.72$. For the ABC algorithm; $N_{bee} = 30$, $BC_{Max} = 200$, $\alpha = 1$, $\beta = 10$ and $\lambda = 0.99$. For the ACO algorithm: The number of ants $m = 30$, $\alpha = 1$, $\beta = 3.5$, $\tau_{init} = 1$, $Q = 10000$ and $v = 10000$. From Table 4.1 and Fig. 4.4, it is noted that at each dataset, the ACO algorithm performs much better than GA and SA (on average, 0.601 and 0.117 normalized distances, respectively). Compared to the ABC algorithm, the ACO algorithm performs exactly the same as the ABC for the first dataset and better in the second and third datasets (on average, the ACO is better than the ABC algorithm by 0.0471 normalized distances). Hence, for such experiment the ACO algorithm shows significant performance compared to the other algorithms.

Table 4.1: Experimental results to three data sets each contain 30 random cities.

	Algorithm			
Data set	GA	SA	ABC	ACO
1	4.9605	4.3528	4.2166	4.2166
2	5.3121	4.9101	4.8588	4.7823
3	5.0694	4.6256	4.6041	4.5392
Average Tour	**5.114**	**4.6295**	**4.5598**	**4.5127**

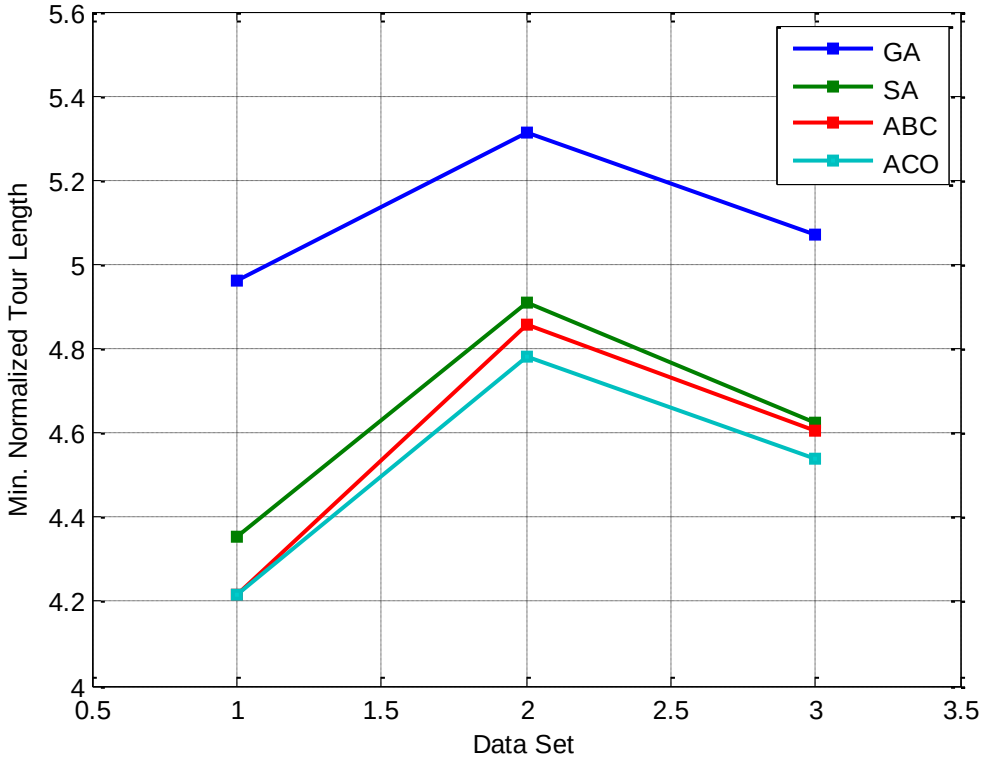


Figure 4.4: Minimum normalized distance obtained using GA, SA, ABC and ACO for the three data sets.

However, to be more satisfied with the performance of ACO in TSP compared to the others, three new datasets containing 50, 100 and 150 random cities, respectively, are generated. The experiments were repeated using the same algorithms and the same implementation parameters. The cities positions for each of these data sets are shown in Figs. 4.5 - 4.7, respectively. The obtained results are shown in the Table 4.2 for all the algorithms.

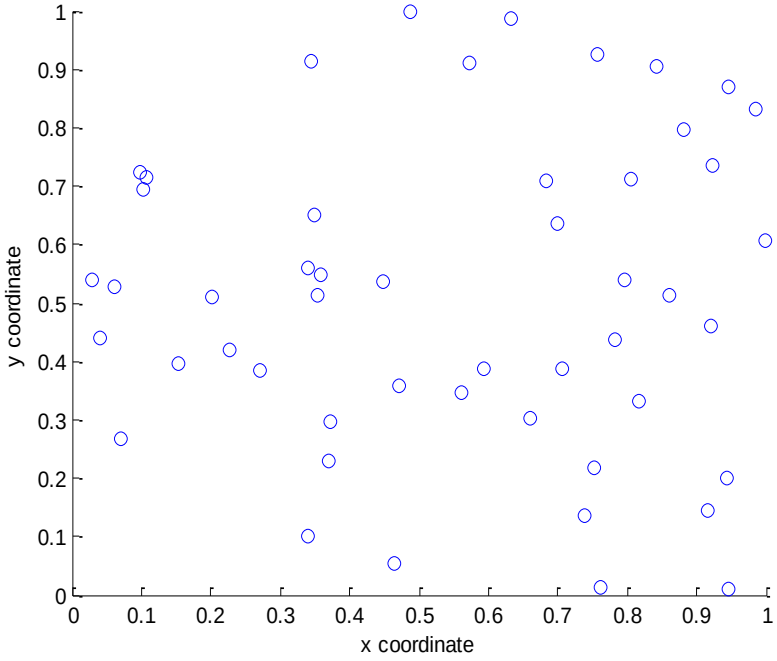


Figure 4.5: Random cities positions of data set 1 containing 50 cities

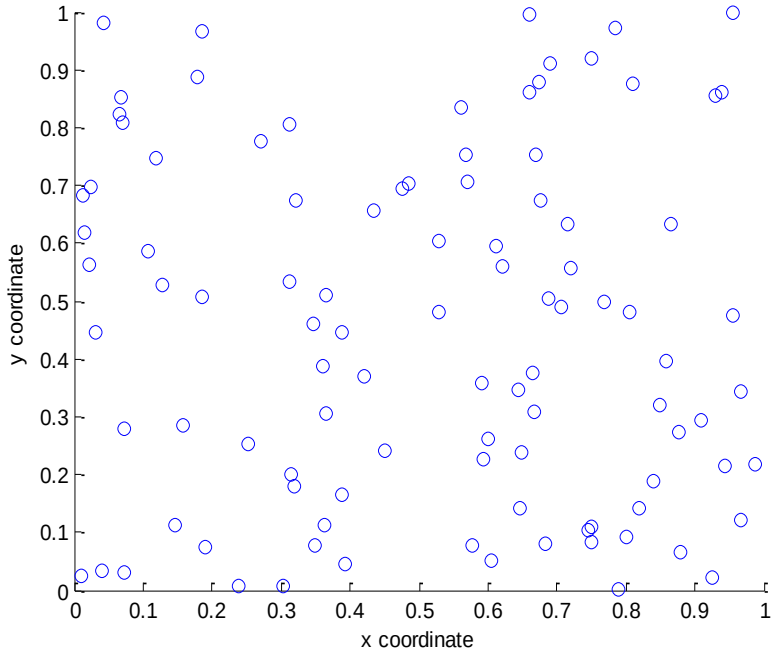


Figure 4.6: Random cities positions of data set 2 containing 100 cities

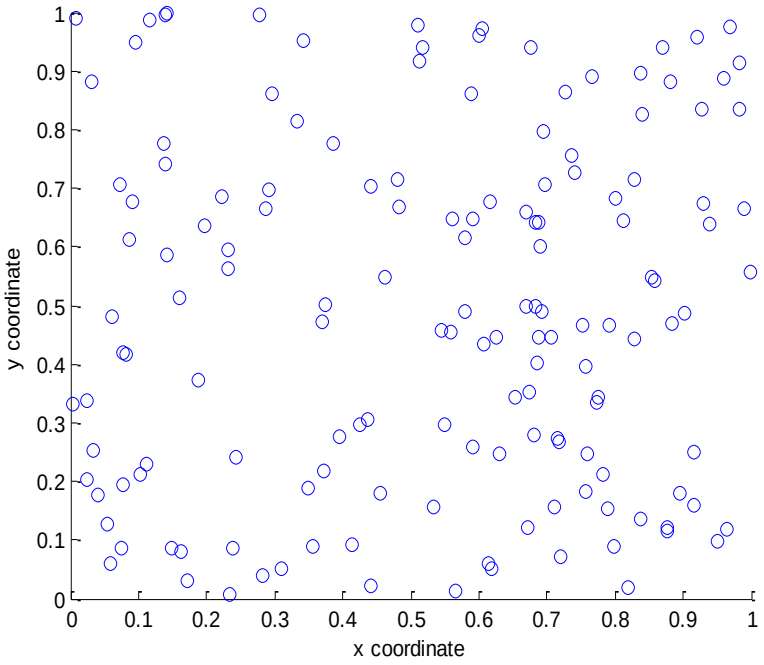


Figure 4.7: Random cities positions of data set 3 containing 150 cities

Table 4.2: Results to three data sets contain 50,100 and150 cities.

	Algorithm			
Data set	GA	SA	ABC	ACO
50 cities	8.4765	6.9529	6.3625	5.9343
100 cities	19.5233	14.1877	8.6425	8.4165
150 cities	33.9455	22.5241	10.2845	10.2166

From Table 4.2 and Fig. 4.8, it is noticed that the performances of the GA and SA algorithms deteriorates. This is due to the optimum path followed is local and not global. However, the ABC and ACO algorithms provide very robust results and very shorter paths than the others with the best performance for the ACO algorithm.

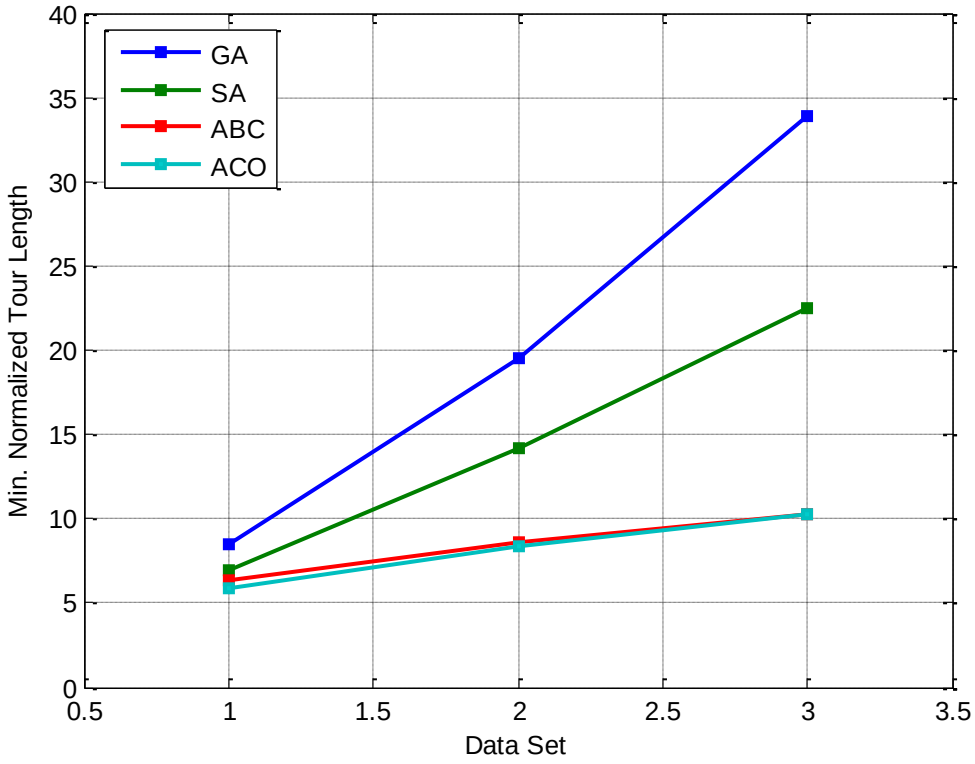


Figure 4.8: Minimum normalized distance obtained using GA, SA, ABC and ACO for the 50, 100 and 150 data sets.

Because the ACO algorithm provides the best performance among all the algorithms, we thought of expanding it into the 2-D case and apply it to the image edge detection problem.

4.3. Edge Detection

In this section, Lena, Camera-man and Barbara images are used to test the performances of the 2-D ACO and the proposed algorithms. The resolutions of all images used are of 512×512 pixels. The parameters used for implementing the conventional ACO and the two proposed algorithms were the same and are: $\alpha=1$, $\beta=0.1$, $\rho=0.1$, $\psi=0.05$, $\tau_{init}=0.1$, $N=10$, $K=4$, $M=40$ and $e=0.1$.

4.3.1. Discrete Wavelet Transform-based ACO for Edge Detection

This section presents performance comparisons between the first proposed algorithm and conventional ACO algorithms under different test images condition. Lena and Camera-man images were used in this section.

In the first experiment, image edge detection using both algorithms is applied to the original Lena image. Fig. 4.9 (b) shows that the conventional ACO algorithm is capable to detect edges very efficiently. However, from Fig. 4.8 (c) we notice that, the first proposed algorithm detects the same edges but denser, which enables us to see some edges that may not be seen using the conventional ACO. This could be very clear at the left side of the image and the nose.

(a)

(b) (c)

Figure 4.9: (a) Original test Image (Lena), (b) Detected edges using ACO [38] and (c) Detected edges using the first proposed algorithm.

In order to see the effect of noise on the performance of the algorithms, in the second experiment, a normalized additive white Gaussian noise (AWGN) with zero mean and variance $\left(\sigma^2 = 0.02\right)$ is added to the images. Image with noise is shown in Fig. 4.10 (b). Figs. 4.10 (c) and (d) show the edges detected by the conventional ACO and the first proposed algorithm, respectively. It is obvious that the first proposed algorithm provides clearer detected edges than the conventional ACO algorithm. This is due to the ability of the DWT in suppressing the noise before detecting the edges.

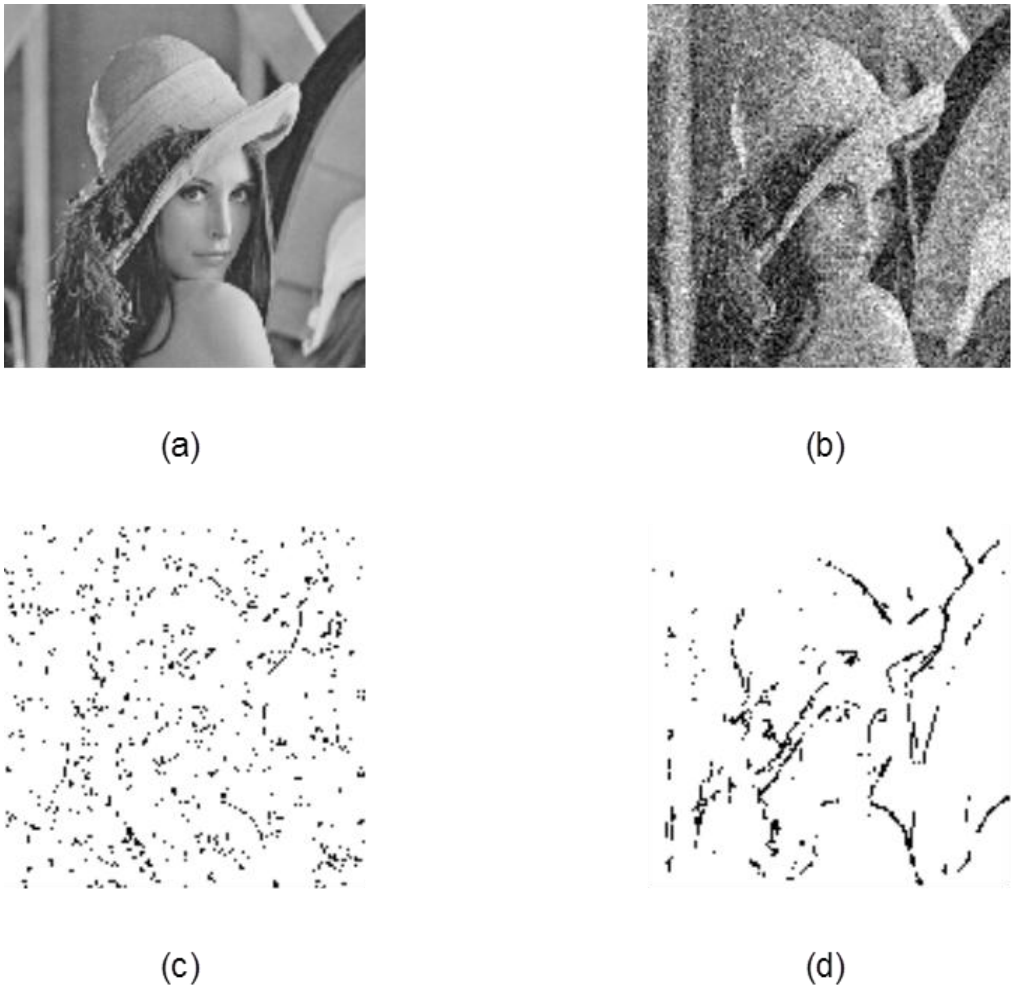


Figure 4.10: (a) Original test Image, (b) Image with noise $\left(\sigma^2 = 0.02\right)$, (c) Detected edges using ACO and (d) Detected edges using the first proposed algorithm.

In order to see the effect of the noise amount on the performances of the algorithms, AWGN with zero mean and variance $\left(\sigma^2 = 0.05\right)$ is added to the normalized images. Fig. 4.11 (b)

shows the image after the noise is added. Fig. 4.11 (c) shows the edge detection result using the conventional ACO algorithm and Fig. 4.11 (d) shows the result of the first proposed algorithm. It is very clear from the results that the conventional ACO algorithm is not capable of detecting the edges of the image buried in noise. However, the first proposed algorithm is still capable of detecting the edges from the noisy image as can be seen from Fig. 4.11 (d).

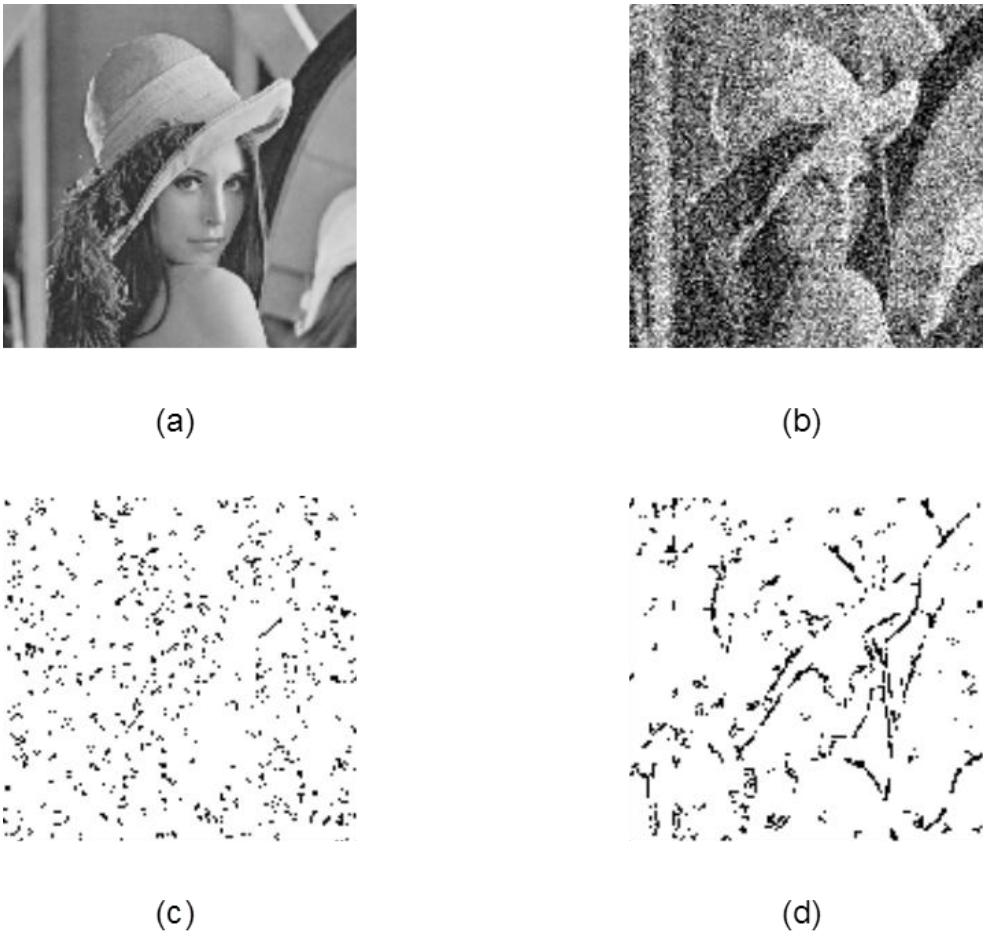
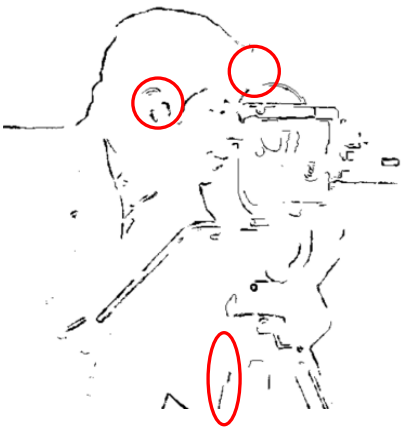


Figure 4.11: (a) Original test Image, (b) Image with noise $\left(\sigma^2 = 0.05\right)$, (c) Detected edges using ACO and (d) Detected edges using the first proposed algorithm.

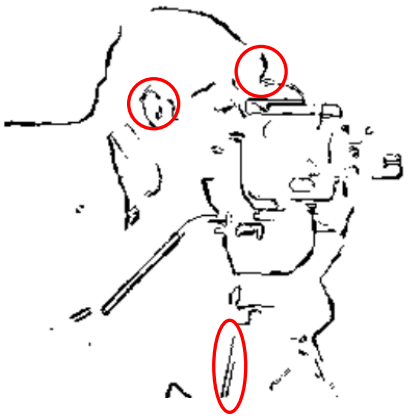
In the second part of this section, we try to show the performances of the conventional ACO and the first proposed algorithm in different images. For this purpose, the Camera-man image is used in this section. From Figs. 4.12 (b) and (c) we noticed that the first proposed algorithm is capable of detecting edges which are not detected by the conventional ACO.

This is very clear at the ear-side of the man and at the front-side of the camera. In addition to that, the edges detected by the first proposed algorithm are denser than those detected by the conventional ACO.

(a)



(b)



(c)

Figure 4.12: (a) Original test Image (Camara-man), (b) Detected edges using ACO and (c) Detected edges using the first proposed method.

The second experiment is repeated with the same parameters, but using the Camera-man image. It is clear from Figs. 4.13 (c) and (d) that the conventional ACO almost fails in detecting the edges of the image where the first proposed algorithm is capable of detecting the edges successfully.

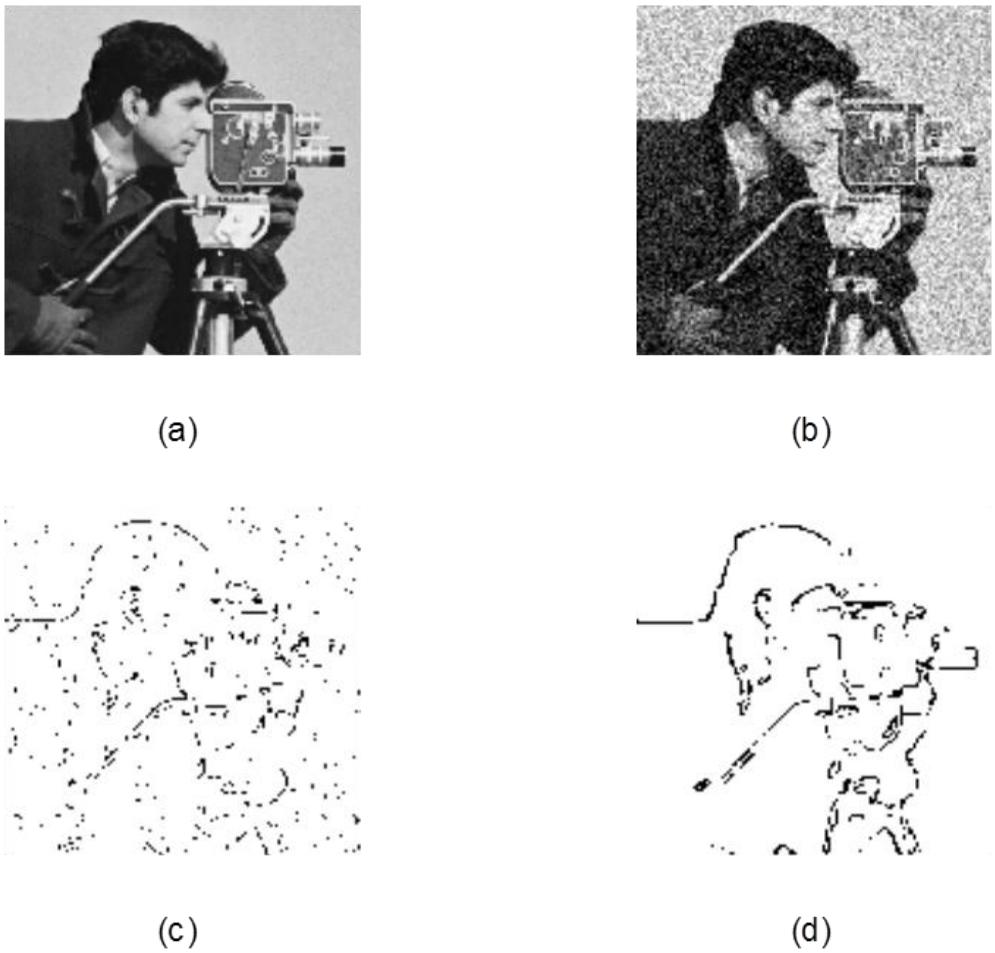


Figure 4.13: (a) Original test Image, (b) Image with noise $\left(\sigma^2 = 0.02\right)$, (c) Detected edges using ACO and (d) Detected edges using the first proposed algorithm.

Finally, from Figs. 4.14 (c) and (d) we noticed that, the amount of the additive noise has almost no effect on the first proposed algorithm. However, this noise amount makes the detection capability of the conventional ACO worse.

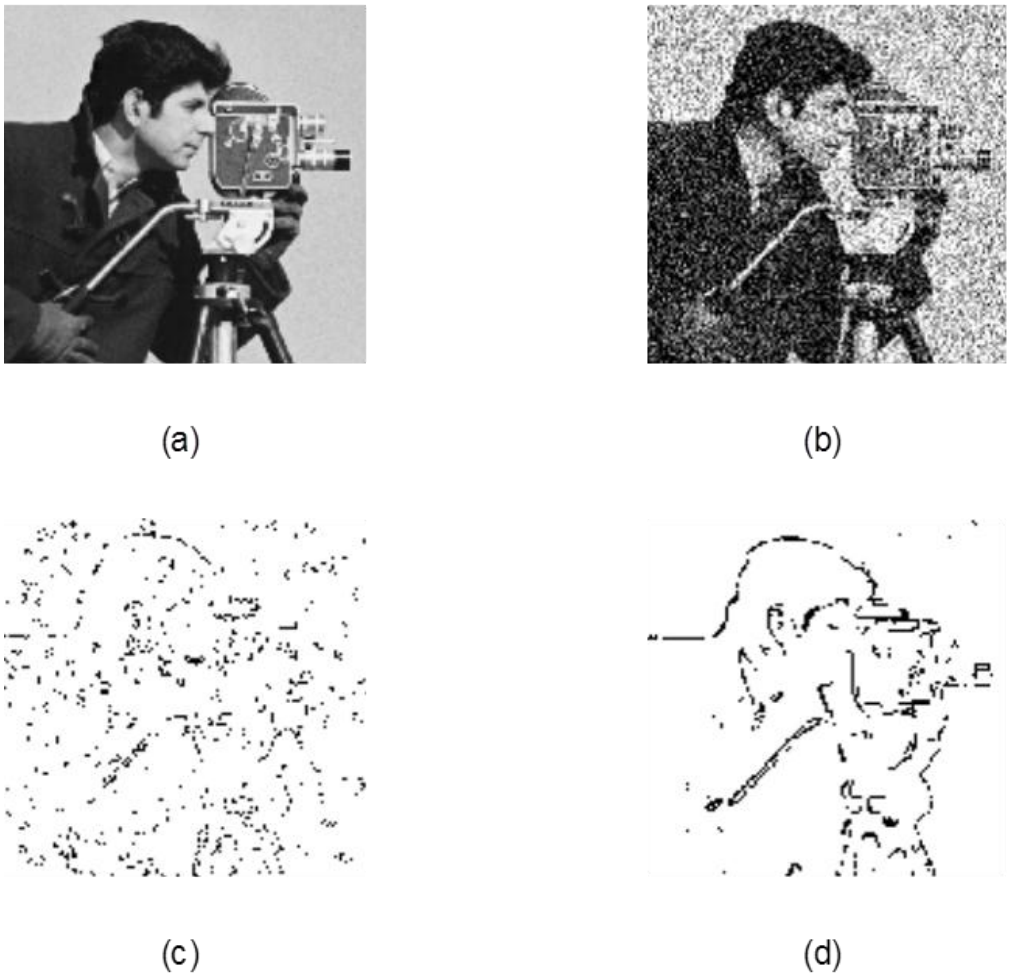


Figure 4.14: (a) Original test Image, (b) Image with noise $\left(\sigma^2 = 0.05\right)$, (c) Detected edges using ACO and (d) Detected edges using the first proposed algorithm.

4.3.2. Summary and Discussions

To measure the performance of any edge detection algorithm, usually a comparison is made between its edge map results with its ground truth image. This can be achieved through many ways that includes; the number of correctly detected edge pixels, called true positive (TP), the number of pixels wrongly classified as edge pixels, called false positive (FP), the number of edge pixels not detected as edge pixels, called false negative (FN), [39]. These ways of performance measurements can be defined mathematically, as:

The percentage of edge pixels that were detected correctly is given in Eq. (4.1), [40].

$$p_{cd} = \frac{TP}{\max(N_I, N_B)} \tag{4.1}$$

where N_I and N_B represent the number of edge points of the ideal image and the number of edge points of the detected image, respectively.

The percentage of edge pixels that were not detected is given in Eq. (4.2).

$$p_{nd} = \frac{FN}{\max(N_I, N_B)} \tag{4.2}$$

The percentage of edge pixels that were not edges but wrongly detected as edges is given in Eq. (4.3).

$$p_{wd} = \frac{FP}{\max(N_I, N_B)} \tag{4.3}$$

Pratt's figure of merit (FOM) is another important measure for assessing the performance of edge detection algorithms. FOM measures the distance between all pairs of points corresponding to quantify, with precision, the difference between the contours [39]. The FOM is defined in Eq. (4.4) [41, 42] and assesses the similarity between two contours [39].

$$FOM = \frac{1}{\max(N_I, N_B)} \sum_{1}^{N_B} \frac{1}{1 + w \times d_i^2} \tag{4.4}$$

where w is a scaling constant that is adjusted to penalize edge points that are detected but offset from exact or true position and its optimal value is $\frac{1}{9}$ as given by Pratt [43], and d is the distance of separation of an actual edge point along a line normal to a line of ideal edge points [40, 44].

The value of FOM ranges between 0 and 1. The larger the value of FOM, the better the performance [45].

In testing the performance of edge detection algorithm, visual method is another method different from objective method. In visual method usually an edge image is evaluated by a group of people and the average score is used as an index of quality [41].

In our work, we used the visual method up to this point and from now on we will use the objective (evaluation) method to test the performance of our algorithms. FOM is used among the objective techniques because it is a widely used objective standard to rate the quality of edge detection algorithms [42].

However, in testing the performances of the proposed algorithms, because the ground truth images are very rear and difficult to get, we used the Canny results of transformed Lena and Camera-man images to be our ground truth images for testing our proposed DWT-Based ACO for edge detection and Canny results of Lena and Barbara images without transform for the conventional ACO, Prewitt and Sobel because, up to our knowledge, Canny gives the best results. The ground truth images used for testing the proposed algorithm are shown in Fig. 4.15.

(a) (b)

Figure 4.15: Ground truth images obtained using Canny and DWT (a) Lena, (b) Camera-man.

From Table 4.3 and Fig. 4.16, it is noted that at each normalized AWGN variance, the first proposed algorithm performs much better than ACO, Sobel and Prewitt. Compared to the Canny operator, our proposed algorithm performs less than Canny in the absence of noise and much better than Canny when the image is corrupted by noise.

Table 4.3: Figure of Merit results for Lena Image.

	Edge Detectors				
Normalized AWGN variance	Canny	Sobel	Prewitt	ACO	First Proposed Algorithm
0	1	0.9419	0.9416	0.9327	0.9435
0.1	0.9738	0.9266	0.9264	0.9655	0.9821
0.2	0.9725	0.9234	0.9232	0.9665	0.9915
0.3	0.9721	0.9220	0.9219	0.9676	0.9912
0.4	0.9716	0.9211	0.9210	0.9672	0.9912
0.5	0.9714	0.9208	0.9206	0.9681	0.9908

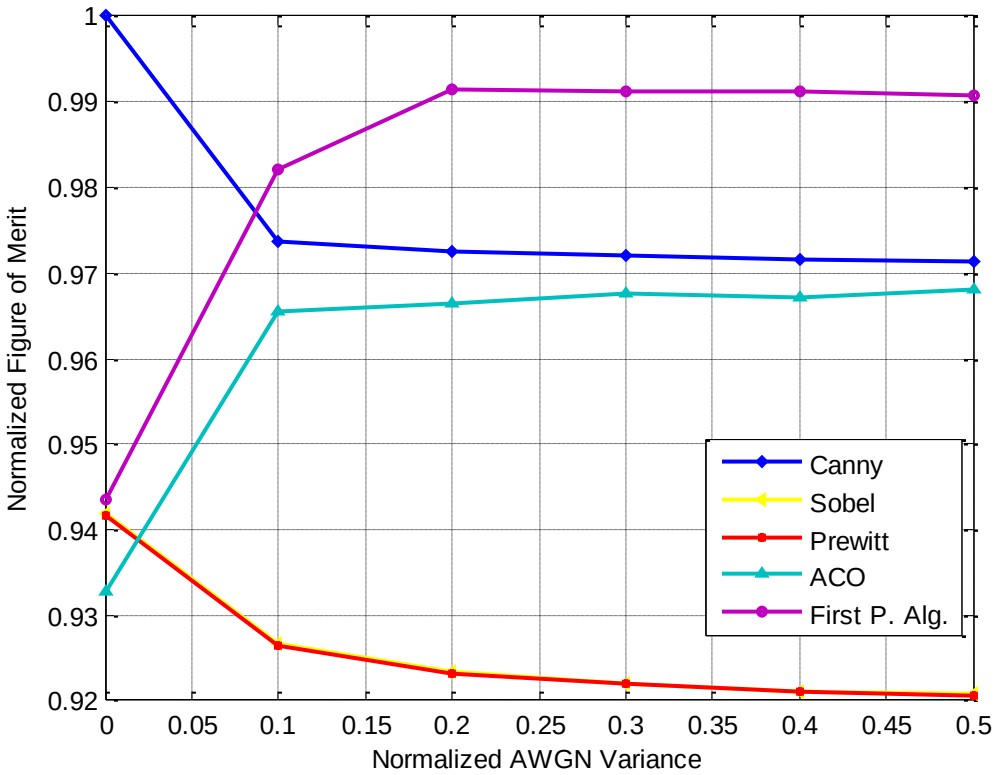


Figure 4.16: Figure of merit for Lena Image.

In the second part of this section, we try to test the performance of our first proposed algorithm on Camera-man image. From Table 4.4 and Fig. 4.17, it is noticed that our first proposed algorithm performs less than Sobel, Prewitt and Canny but better than the conventional ACO in the absence of noise. However, when the image is buried with noise, our first proposed algorithm performs much better than the others.

Table 4.4: Figure of Merit results for Camera-man Image

	Edge Detectors				
Normalized AWGN variance	Canny	Sobel	Prewitt	ACO	First Proposed Algorithm
0	1	0.9557	0.9556	0.9511	0.9531
0.1	0.9759	0.9493	0.9493	0.9817	0.9848
0.2	0.9741	0.9448	0.9449	0.9847	0.9901
0.3	0.9733	0.9417	0.9421	0.9856	0.9927
0.4	0.9728	0.9404	0.9406	0.9866	0.9924
0.5	0.9722	0.9397	0.9396	0.9852	0.9924

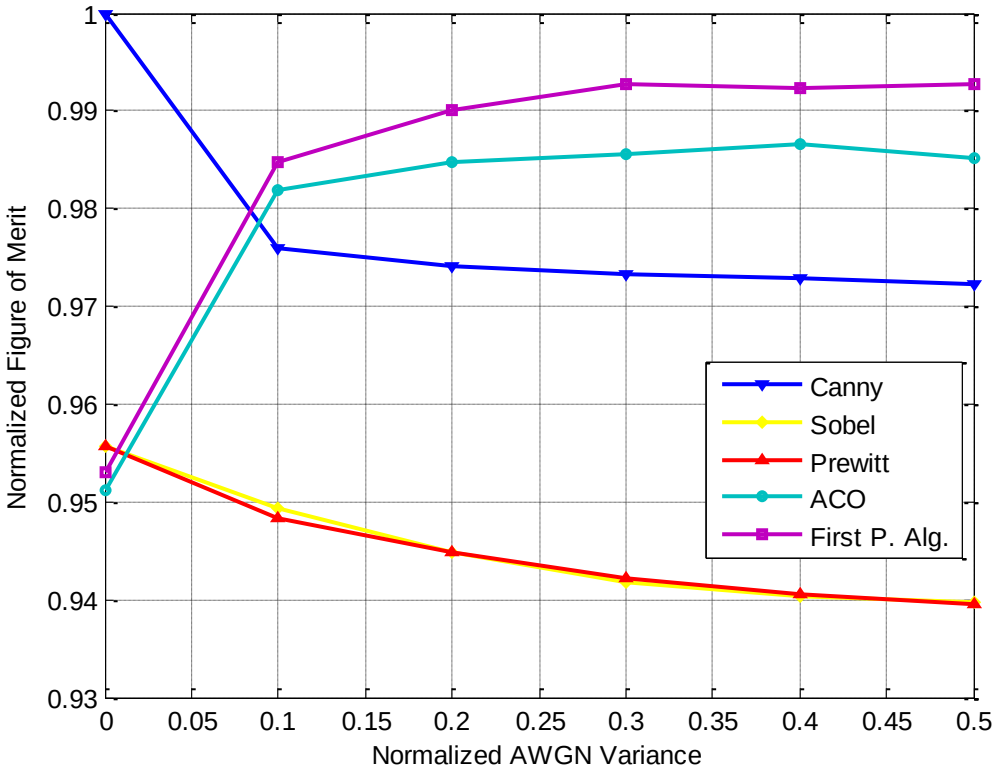


Figure 4.17: Figure of merit for Camera-man Image

4.3.3. DWT Sub-Band Fusion using ACO for Edge Detection

This section shows the performance of the second proposed algorithm compared to that of the conventional ACO. Lena, Barbara and Camera-man images were used to test the performances of both algorithms. Even though the second proposed algorithm needs more time than the conventional ACO, it provides much denser edges is capable of detecting edges that are not detected by the conventional ACO algorithm. This algorithm can be used in applications were the detected edges are more important than the time.

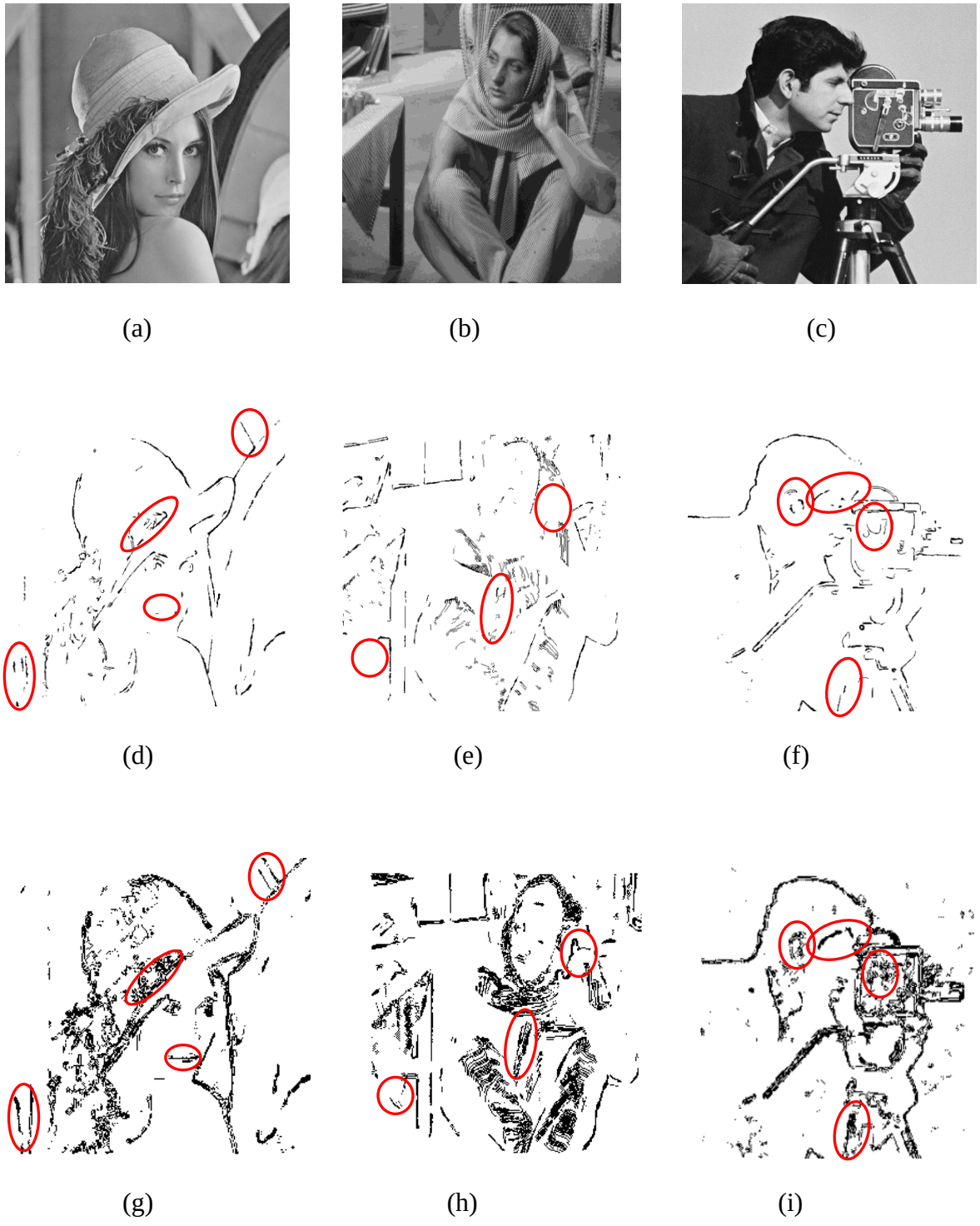


Figure 4.18: (a) Lena image (b) Barbara image (c) Camera-man image (d) Lena edges using ACO (e) Barbara edges using ACO (f) Camera-man edges using ACO (g) Lena edges using the second proposed method. (h) Barbara edges using the second proposed method (i) Camera-man edges using the second proposed method.

Like the first proposed algorithm, the second proposed algorithm was also tested using Pratt's FOM. Lena and Barbara images were used to test the performance of this algorithm. From Table 4.5 and Fig. 4.19, it is very clear that the performance of our second proposed algorithm is better than those of the ACO and the first proposed algorithm.

Table 4.5: Figure of merit results

	Edge Detector Algorithms		
Test Images	ACO	First Proposed Algorithm	Second Proposed Algorithm
Lena	0.9339	0.9458	0.9475
Barbara	0.9237	0.9391	0.9386

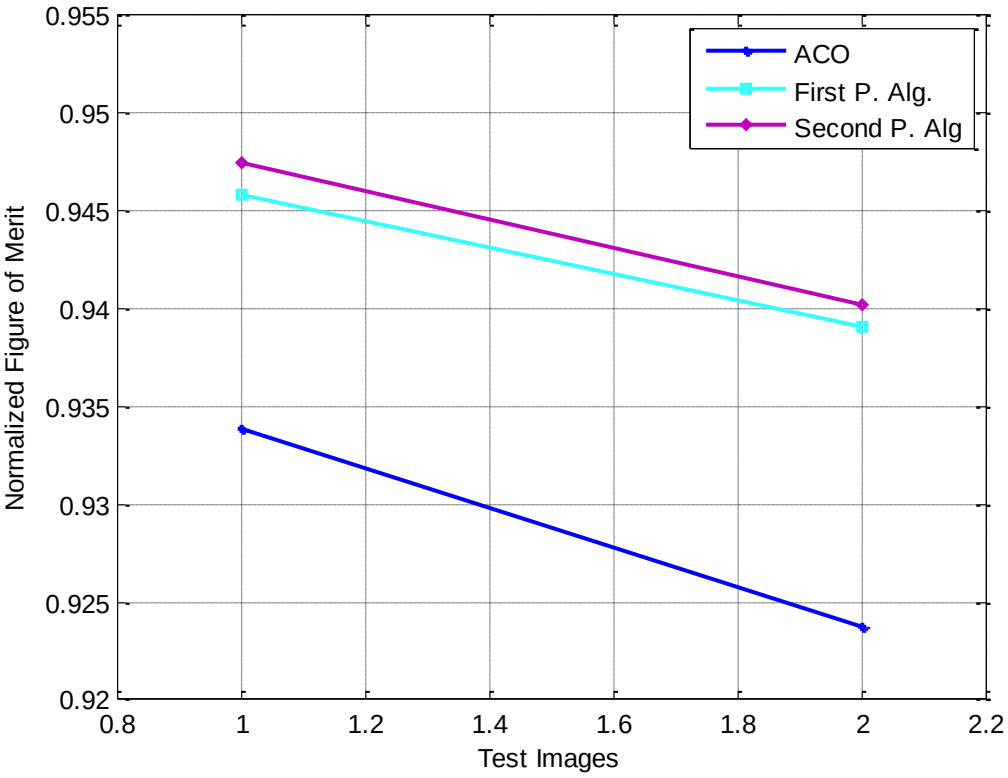


Figure 4.19: Figure of merit for Lena and Barbara images.

CHAPTER FIVE

CONCLUSIONS AND FUTURE WORKS

5.1. Conclusions

In this thesis, the performances of four metaheuristic algorithms (GA, SA, ABC and ACO) were tested in solving TSP using various positions and number of cities. The results obtained in all the situations show that ACO performs better than others. For this reason ACO is selected and applied in the 2-D case for edge detection problem. However, after ACO is successfully applied for the edge detection problem, two algorithms were proposed to improve the results. The first method was based on DWT and ACO. The DWT is applied to the image as a preprocessing stage before conventional ACO algorithm. The edge detection results obtained using this technique show substantial improvements, more especially, when the original image is noisy.

Another approach is proposed to enhance edge detection results more, by using each of the four sub-images obtained, by decomposing the original image using 2D-DWT and finally recombining the results by 2D-IDWT. The proposed algorithm is capable of detecting some edges that are not detected by the conventional ACO and it also provides a denser edge detection performance.

The performances of the two proposed approaches were measured using a widely used standard objective approach (Pratt's figure of merit). The first proposed algorithm performance was tested on Lena and Camera-man images under different noise conditions. The algorithm is found to perform well in edge localization even with high noise powers. The second proposed algorithm performance was tested on Lena and Barbara images. The results obtained show that the second proposed algorithm performs better than the ACO and first proposed algorithm.

5.2. Future Works

In this work, DWT is used with the conventional ACO to propose two algorithms for improving the performance ACO in edge detection. As a future work, the followings may be considered:

- In all the experiments for the 1-D case, it is noticed that the performances of the ABC were close to ACO. Hence, it can be applied for the 2-D case as well, and the results may be compared to that of the ACO.
- The GA and SA algorithms may be applied to the 2-D case for edge detection problem and their performances might be compared to those of the proposed algorithms.

References

[1] D. Whitley, “An Overview of Evolutionary Algorithms: Practical Issues and Common Pitfalls,” *Information and Software Technology*, pp. 817-831, 2001.

[2] G. Deng and Z. H. Ming Tang, “Research in the Performance Assessment of Multi-objective Optimization Evolutionary Algorithms,” *IEEE International Conference on Communications, Circuits and Systems (ICCCAS)*, Kokura, pp. 915-918, 2007.

[3] D. E. Moriarty, A. C. Schultz and J. J. Grefenstette, “Evolutionary Algorithms for Reinforcement Learning,” *Journal of Artificial Intelligence Research*, vol. 11, pp. 241-276, 1999.

[4] G. Jones, “Genetic and evolutionary algorithms,” *Encyclopedia of Computational Chemistry, John Wiley and Sons*, 1998.

[5] K. Weicker, “Evolutionary algorithms and dynamic optimization problems,” *Berlin: Der Andere Verlag*, 2003.

[6] Y. Xin-She, “Engineering optimization: an introduction with metaheuristic applications,” *John Wiley & Sons*, 2010.

[7] Y. Yang, H. Dai and H. Li, “Adaptive Genetic Algorithm with Application for Solving Travelling Salesman Problems,” *IEEE International Conference on Internet Technology and Applications*, Wuhan, pp. 1-4, 2010.

[8] G. Zhao, W. Luo, H. Nie and C. Li, “A Genetic Algorithm Balancing Exploration and Exploitation for the Travelling Salesman Problem,” *IEEE Fourth International Conference on Natural Computation*, Jihan, vol. 1, pp. 505-509, 2008.

[9] L. Zhang, M. Yao and N. Zhen, “Optimization and Improvement of Genetic Algorithms Solving Travelling Salesman Problem,” *IEEE International Conference on Image Processing and Signal Processing*, Taizhou, pp. 327-332, 2009.

[10] D. Wierstra, T. Schaul, J. Peters and J. Schmidhuber, “Natural Evolution Strategies,” *IEEE World Congress on Computational Intelligence*, Hong Kong, pp. 3381-3387, 2008.

[11] B. Thomas, “Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms,” *Oxford university press*, 1996.

[12] C. Doo-Hyun, "Evolutionary Programming with Accumulated Evolution Information," *IEEE Electronics Letters*, vol. 35, issue 10, pp. 808-809, 1999.

[13] W. M. Spears, "The role of mutation and recombination in evolutionary algorithms," PhD diss., George Mason University, 1998.

[14] C. Blum, "Metaheuristic for Group Shop Scheduling," Dipl. Thesis, Iridia University, 2002.

[15] K. S. Lee and Z. W. Geem, "A new Meta-heuristic Algorithms for Continuous Engineering Optimization: Harmony Search Theory and Practice" *Computer Methods in Applied Mechanics and Engineering*, vol. 194, issues 36-38, pp. 3902-3933, 2005.

[16] H. G. Shakouri, K. Shojaee and M. T. Behnam, "Investigation on the Choice of the Initial Temperature in the Simulated Annealing: A Mushy State SA for TSP," *IEEE Mediterranean Conference on Control and Automation*, Makedonia Palace, Greece, pp. 1050-1055, 2009.

[17] K. Shojaee, T. Behnam, G. Shakouri, and M. Rezaei, "Enhancement of SA algorithm by intelligent time schedule," *IEEE Chinese Control Conference (CCC),* pp. 1768-1774, 2010.

[18] P. Selami and A. Mustapha, "A Novel Cloning Template Designing Method by Using an Artificial Bee Colony Algorithm for Edge Detection of CNN Based Imaging Sensors," *Academic Journal on Sensors*, vol. 11, issue 5, pp. 5337-5359.

[19] B. Kamalam and K. Marcus, "A Comprehensive review of Artificial Bee Colony Algorithm," *International Journal of Computer and Technology*, vol. 5, No. 1, pp. 15-28, 2013.

[20] S. Fazli and S. F. Ghiri, "Automatic Circle Detection in Digital Images using Artificial Bee Colony," *International Conference on Advances in Computer and Electrical Engineering (ICACEE),* pp. 21-24, 2012.

[21] M. Dorigo and G. Di Caro, "Ant Algorithms for Discrete Optimization," *MIT Press*, 1999.

[22] S. Thomas, and H. Hoos, "The max-min ant system and local search for combinatorial optimization problems," *Springer* US, pp. 313-329, 1999.

[23] J. Ouyang and Y. Gui-Rong, "A multi-group ant colony system algorithm for TSP," *IEEE International Conference on Machine Learning and Cybernetics*, vol. 1, pp. 117-121, 2004.

[24] E. Hetmaniok, D. Slota, A. Zielonka and R. Witula, "Comparison of ABC and ACO algorithms applied for solving the inverse heat conduction problem," *Springer, Swarm and Evolutionary Computation*, Berlin Heidelberg, pp. 249-257, 2012.

[25] A. Muhammad, I. Bala, M. S. Salman and A. Eleyan, "Discrete wavelet transform-based ant colony optimization for edge detection," *IEEE International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE2013)*, Turkey, pp. 280-283, 2013.

[26] W. Li-Pei, Y. H. Malcolm and C. S. Chong, "A Bee Colony Optimization Algorithm for Travelling Salesman Problem," *IEEE Second Asia International Conference on Modeling and Simulation (AICSM)*, Kuala Lumpur, pp. 818-823, 2008.

[27] G. Laporte, "The travelling salesman problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 231-247, 1992.

[28] M. Dorigo and T. Stutzle, "Ant Colony Optimization," *MIT Press*, 2004.

[29] J. L. Pasquier, I. K. Balich, D. W.Carr and C. Lopez-Martin, "A Comparative Study of three Metaheuristic applied to the Travelling Salesman Problem," *IEEE International Conference on Artificial Intelligence*, pp. 243-254, 2007.

[30] H. S. Neoh and A. Hazanchuk, "Adaptive Edge Detection for Real-Time Video Processing using FPGAs," *Global Signal Processing*, 2004.

[31] R. Maini and Dr. H. Aggrawal, "Study and Comparison of Various Edge Detection Techniques," *International Journal of Image Processing (IJIP)*, vol. 3, Issue (1), pp. 1-12, 2009.

[32] E. Nadernejad, S. Sharifzadeh, and H. Hassanpour, "Edge detection techniques: evaluations and comparisons," *Applied Mathematical Sciences* vol. 2, no. 31, pp. 1507-1520, 2008.

[33] B. A. Veronica, "Ant colony optimization for image edge detection," *PhD diss., Ateneo de Manila University*, 2010.

[34] J. Tian, W. Yu and S. Xie, "An Ant Colony Optimization for Image Edge Detection," *IEEE World Congress on Computational Intelligence*, Hong Kong, pp. 751-756, 2008.

[35] Y. S. Al-Halabi and J. A. Hesham, "New Wavelet-Based Techniques for Edge Detection," *Journal of Theoretical & Applied Information Technology*, vol. 23, no. 1, 2011.

[36] X. Zhang and R. Zhang, "The technology research in decomposition and reconstruction of image based on two-dimensional wavelet transform," *International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 1998-2000, 2012.

[37] A. Muhammad, I. Bala, M. S. Salman and A. Eleyan, "DWT subbands fusion using ant colony optimization for edge detection," *IEEE Signal Processing and Communications Application Conference (SIU)*, pp. 1351-1354, 2014.

[38] P. Agrawal, S. Kaur, H. Kaur and A. Dhiman, "Analysis and Synthesis of an Ant Colony Optimization Technique for Image Edge Detection," *IEEE International Conference on Computing Sciences (ICCS)*, pp. 127-131, 2012.

[39] A. Halder, N. Chatterjee, A. Kar, S. Pal and S. Pramanik, "Edge Detection: A Statistical approach," *International Conference on Electronics Computer Technology (ICECT)*, vol. 2, pp. 306-309, 2013.

[40] B. Chanda, M. K. Kundu and Y. V. Padmaja, "A Multi-Scale Morphologic Edge Detector," *Pattern Recognition, Elsevier*, vol. 31, no. 10, pp. 1469-1478, 1998.

[41] Z. J. Hou and G. W. Wei, "A new approach to edge detection," *The Journal of the Pattern Recognition Society*, vol. 35, pp. 1559-1570, 2002.

[42] Y. Yuan-Hui and C. Chin-Chen, "A new edge detection approach based on image context analysis," *Elsevier Image and Vision Computing*, vol. 24, pp. 1090-1102, 2006.

[43] I. A. Abdou and W. Pratt, "Quantitative design and evaluation of enhancement/ thresholding edge detectors," *IEEE Proceedings*, vol. 67, no. 5, pp. 753-766, 2005.

[44] S. Pande, V. S. Bhadouria and D. Ghoshal, "A study on edge marking scheme of various standard edge detectors," *International Journal of Computer Applications*, vol. 44, no. 9, pp. 33-37, 2012.

[45] K. A. Panetta, E. J. Wharton and S. S. Agaian, "Logarithmic Edge Detection with Applications," *International Journal of Computers*, vol. 3, no. 9, pp. 11-19, 2008.