A

Project Report

On

"Optimization Scheme for Power Transmission in Wireless Sensor Network"

Submitted for partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Electrical Engineering

Session: 2022-23

Supervised by: -
Dr. Ankit Vijayvargiya
Associate Professor
Dept. of Electrical Engineering

Submitted by:
Anshul Kumar Yadav (19ESKEE023)
Akshita Sharma (19ESKEE012)
Anurag Yadav (19ESKEE029)
Ankita (19ESKEE019)
Group: - A/G1/7

Department of Electrical Engineering
Swami Keshvanand Institute of Technology, Management & Gramothan
Ramnagaria, Jagatpura, Jaipur (RAJ.)

Swami Keshvanand Institute of Technology Management and Gramothan, Jaipur

CERTIFICATE

This is to certify that **Anshul Kumar Yadav, Akshita Sharma, Anurag Yadav,** and **Ankita** of VIII semester, B.Tech. (Electrical Engineering) worked on their project entitled "**Optimization Scheme for Power Transmission in Wireless Sensor Network** " under my guidance; is being submitted for the partial fulfilment of the award of the degree of B.Tech. from Rajasthan Technical University, Kota during the session 2022-23.

Date:

Place: Jaipur

Supervisor

Dr. Ankit Vijayvargiya

Associate Professor

Dept. of Electrical Engineering

CANDIDATES DECLARATION

We hereby declare that the project report titled, " **Optimization Scheme for Power Transmission in Wireless Sensor Network** " is our own work conducted under the supervision of **Dr. Ankit Vijayvargiya, Associate Professor, Department of Electrical Engineering, SKIT**. We confirm that:

- This work is done towards the partial fulfilment of the degree of "Bachelor of Technology" at Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur.
- Wherever any part of this report has previously been submitted for a degree or any other qualification at SKIT or any other institution, this has been clearly stated.
- Wherever we have consulted the published work of others, this is always clearly mentioned.
- We have acknowledged all main sources of help.

Signature:	Signature:
Name of Student: Anshul Kumar Yadav	Name of Student: Akshita Sharma
Roll No.:19ESKEE023	Roll No.:19ESKEE012
Signature:	Signature:
Name of Student: Anurag Yadav	Name of Student: Ankita
Roll No.:19ESKEE029	Roll No.:19ESKEE019

Date

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any work will be incomplete unless we mention the names of the people who made it possible. We express our sincere gratitude to our project supervisor **Dr. Ankit Vijayvargiya, Associate Professor, Department of Electrical Engineering, Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur**, who spared his valuable time from busy schedules and extended valuable guidance, expertise, and vision to guide us through this project.

We sincerely acknowledge our project in-charge **Mr. Ankush Tandon, Associate Professor, Department of Electrical Engineering, Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur**, for his technical insights, support for literature, critical review, constructive criticism, and above all, the moral support provided us at all stages of this endeavour, without which it would not have been possible to prepare this project.

Much appreciation is due to **Dr. Sarfaraz Nawaz, Head, Department of Electrical Engineering, and Dr. Ramesh Kumar Pachar, Principal, Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur** for providing us with moral support and infrastructure to go ahead with our project work. We wish to thank the other faculty members of Electrical engineering who gave precious guidelines for our project. Their advice, suggestions, and questions will continue to enhance our knowledge and understanding. We thank our friends who have contributed directly or indirectly by giving valuable suggestions

Anshul Kumar Yadav (19ESKEE023)
Akshita Sharma (19ESKEE012)
Anurag Yadav (19ESKEE029)
Ankita (19ESKEE019)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Energy efficiency is critical for prolonging the network's lifetime in energy-constrained wireless sensor networks. As the Internet of Things (IoT) becomes real, several problems are inherent to energy-constrained nodes and mesh networks The issues addressed for WSNs include optimal node deployment, node localization, energy-aware clustering and data aggregation. Most of these issues can be modelled as optimization problems, allowing metaheuristic approaches to be used in finding their solution.

The Optimization process aims to determine the best possible solution for the given problem. The optimization algorithm helps attain results by maximizing or minimizing the parameters in the constraints of a given problem. Different types of optimization algorithms are being developed and used to solve complex problems more accurately. In recent years, these algorithms have proved their use in real-world problems.

There are three subsystems that draw energy from the batteries - the sensing unit, the processor, and the communication unit. Moreover, there are several wireless node events that can be adjusted to mitigate wasting power. Namely, they are packet overhead, idle listening, overhearing, over emitting, collisions, and state transitions. The goal of the present introductory research is to avoid state transitions by finding the optimal transmission power for each node in the network, in order to fully connect sensor nodes in a single cluster

This report presents an overview of how optimization is helping the wireless sensor network (WSNs) to attain better efficiency, fewer losses, and help derive better values. WSNs enable a domain of rich application and research. Isolated sensors can sense a dynamic process while sending measured data to a central base station through a common channel. Using metaheuristic approaches like a grey wolf and particle swarm optimizer in the case of data transmission operation to solve the minimization problem has led to the optimum solution. Also, the results derived from these approaches are then compared with other optimization-based algorithms.

CHAPTER 1

Introduction

1.1 Introduction to optimization algorithm:

With the development of society, people will face more and more complex problems. However, solving a class of complex problems is an essential requirement for promoting social development. Although many traditional numerical and analytical methods have carried out relevant analysis research, some deterministic methods cannot provide a fitting solution to solve several challenging problems with non-convex and highly non-linear search domains since the complexity and dimensions of these problems grow exponentially. Optimizing the problems by applying some deterministic methods, such as the Lagrange and Simplex methods, requires some initial information about the problem and complicated computations. Thus, exploring global optimum solution problems using such methods for those levels of problems is not always possible or feasible. Therefore, developing an efficient method to solve increasingly complex optimization problems is still urgent. Optimization methods can have multiple forms and formulations, maybe no limit in form, and what they essential for them in stochastic class is a core for exploration and a core for exploitation, which can be utilized to deal with those forms of problems, such as multi-objective optimization, fuzzy optimization, robust optimization, memetic optimization, large scale optimization, many-objective optimization methods, and single-objective optimization. One common optimization method, named swarm intelligence (SI) algorithms [1], is swarm-based optimization based on the evolution of an initial set of agents and attraction of agents towards better solutions, which, in an extreme case, is the optimum solution and avoids locally optimal solutions. The swarm intelligence optimization algorithm has intelligent characteristics such as self-adaptation, self-learning, and self-organization and is convenient for large-scale parallel computing. In recent years, some classes of swarm-based optimization algorithms have been applied as simple and reliable methods for realizing the solutions to problems in both the computer science field and industry. Numerous researchers have demonstrated that swarm-based

optimization is very promising for tackling many challenges. Some algorithms employ methods that mimic natural evolutionary mechanisms and basic genetic rules, such as selection, reproduction, mutation, and migration. One of the most popular evolutionary methods is Holland's genetic algorithm (GA) [2]. With its unique three core operations of crossover, variation, and selection, GA has achieved outstanding performance in many optimization problems, such as twice continuously differentiable NLP problems, predicting production, and neural architectures searching. Other well-regarded evolutionary algorithms include differential evolution (DE) [3] and evolutionary strategies (ES) [4]. This kind of evolutionary algorithm simulates the way of biological evolution in nature and has strong adaptability to problems. Moreover, the rise of deep neural networks (DNN) [5] in recent years has made people pay more attention to how to design neural network architecture automatically. Therefore, network architecture search (NAS) [6] based on evolutionary algorithms has become a hot topic. Some methods are motivated by physical laws, such as simulated annealing (SA) As one of the most well-known methods in this family presented by Kirkpatrick, et al., SA simulates the annealing mechanism utilized in physical material sciences. Also, with its excellent local search capabilities, SA can find more potential solutions to many engineering problems than other traditional SI algorithms. One of the latest well-established methods is the gradient-based optimizer (GBO)1, which considers Newtonian logic to explore suitable regions and achieve a global solution. The method has been applied in many fields, including feature selection and parameter estimation of photovoltaic models. Most swarm methods mimic the equations of particle swarm optimization (PSO) [7] by varying the basis of inspiration around the collective social behaviours of animal groups. Particle swarm optimization (PSO) is one of the most successful algorithms in this class, which was inspired by birds' social and individual intelligence when flocking. In detail, PSO has a few parameters that need to be adjusted, also, unlike other methods, PSO has a memory machine, and the knowledge of particles with better performance can be preserved, which can help the algorithm find the optimal solution more quickly. Currently, PSO has taken its place in the fields of large-scale optimization problems feature selection, single-objective optimization problems, multi-

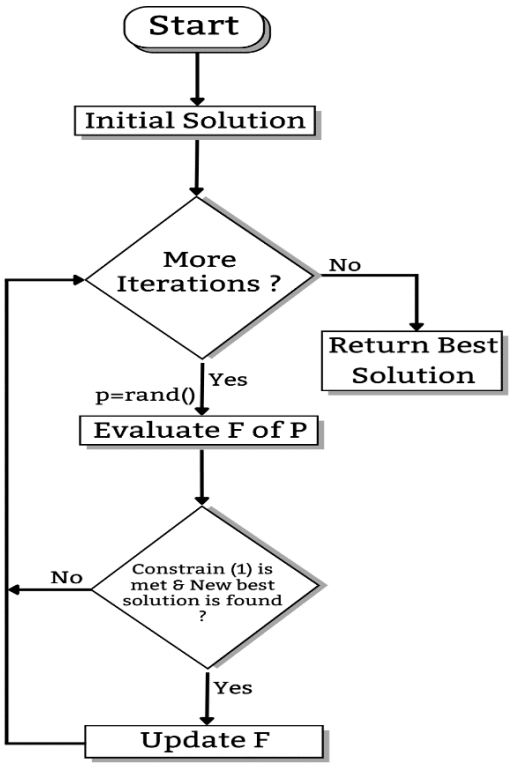


Fig 1.1: - Flowchart of any optimization algorithm

objective optimization problems and high-dimensional expensive problems Ant colony optimization (ACO) [8] is another popular approach based on ants' foraging behaviour. In particular, the concept of pheromone is a major feature of ACO. According to the pheromone secreted by the ants in the process of searching for food, it can help the population to find a better solution at a faster rate. As soon as ACO was proposed, it was applied to the traveling salesman problem of 3 spots and some complex optimization problems and achieved satisfactory results. While these optimization methods can solve various challenging and real optimization problems, the No Free Lunch (NFL) theorem authorizes researchers to present a new variant of methods or a new optimizer from scratch. This theory states that no optimization method can work as the best tool for all problems. Accordingly, one algorithm can be the most suitable approach to solve several problems but is incompetent for other optimization problems. Hence, it can be declared that this theory is the basis of many studies in this field.

A new optimizer (INFO) [9] by modifying the weighted mean method and updating the vectors' position, which can help form a more robust structure. In detail, updating rules, vector combining, and local search are the three core processes of INFO. Unlike other methods, the updating rule based on the mean is used to generate new vectors in INFO, thus accelerating the convergence speed. In the vector combination stage, two vectors acquired in the vector update stage are combined to produce a new vector for improving local search ability. This operation ensures the diversity of the population to a certain extent. Taking into account the global optimal position and the mean-based rule, a local operation is executed, which can effectively improve the problem of INFO being vulnerable to local optimal. This work's primary goal was to introduce the above three core processions for optimizing various kinds of optimization cases and engineering problems, such as structural and mechanical engineering problems and water resources systems. The INFO algorithm employs the concept of weighted mean to move agents toward a better position. This main motive behind INFO emphasizes its performance aspects to potentially solve some of the optimization problems that other methods cannot solve. It should be noted that there is no inspiration part in INFO [9], and it is tried to move the field to go beyond the metaphor

1.2 History:

The field of Optimization starts its gradual perfection in the mid-19th century. Basically, it was derived from the word optimize, which means to make the best or most of." Therefore, optimization refers to the process of finding something as best as it can be. Optimization started in the field of economics by F.Y Edgeworth & Vilfredo Pareto both working at the same time. Edgeworth in 1881 at King's College, London was the first to find an optimum for multivariable economic problems. Pareto had a degree in Civil Engineering from the University of Turin, Italy. While working in Florence, Italy as a civil engineer, he was one of the first to apply mathematical tools to economic problems. Pareto developed the "Pareto Optimum" in 1906. Around 1971 Pareto's work was translated into English which led to the development of many methods in the field of Engineering and Mathematics for multi-objective optimization.[1] J. Farkas

proved the Karush- Kuhn-Tucker theorem using his lemma in 1902. The first book on optimization, "Theory of Minima and Maxima", was published by H. Hancock in 1917. L. V. Kantorovich in 1939 presented a Linear Programming model and also the algorithm to solve it, he and T. C. Koopmans were awarded the Nobel Memorial prize in Economics for their work on linear programming algorithm.[2] J. Von Nueman & O. Morgenstern used the idea of Dynamic programming to solve sequential decision problems in 1944. H. W. Kuhn and A. W. Tucker reinvented optimality conditions for non-linear problems in 1951, F. John (1948) and W. Karush (1939) had presented similar ideas. In 1951, Portfolio theory was proposed by H. Markowitz based on quadratic optimization. He also received Nobel Memorial prize in Economics in 1990. R. Bellman in 1957 presented the optimality principle. In 1960, Zoutendijk presented many methods to generalize the Simplex method in non-linear programming. Wolfe, Powell, and Rosen also developed similar methods. Narendra K. Karmarkar developed one of the first polynomial time algorithms known as the "Interior Point Method" in 1984.[2] After 80's, as computers became faster and more efficient, optimization algorithms were started to be applied to large-scale problems. In 90's Interior point methods expanded to semi definite optimization.[2] As computers and programming became mainstream, a lot of different algorithms based on different methodology began

to boom.[2] The previous studies on optimization methods present this research's primary motivation. Generally, evolutionary algorithms are classified into two types: single-based and population-based algorithms. In the first case, the algorithm's search process begins with a single solution and updates its position during the optimization process. The most well-known single-solution-based algorithms include simulated annealing (SA), tabu search (TS), and hill-climbing. These algorithms allow easy implementation and require only a small number of function evaluations during optimization. However, the disadvantages are the high possibility of trapping in local optima and failure to exchange information because these methods have only a single trend.

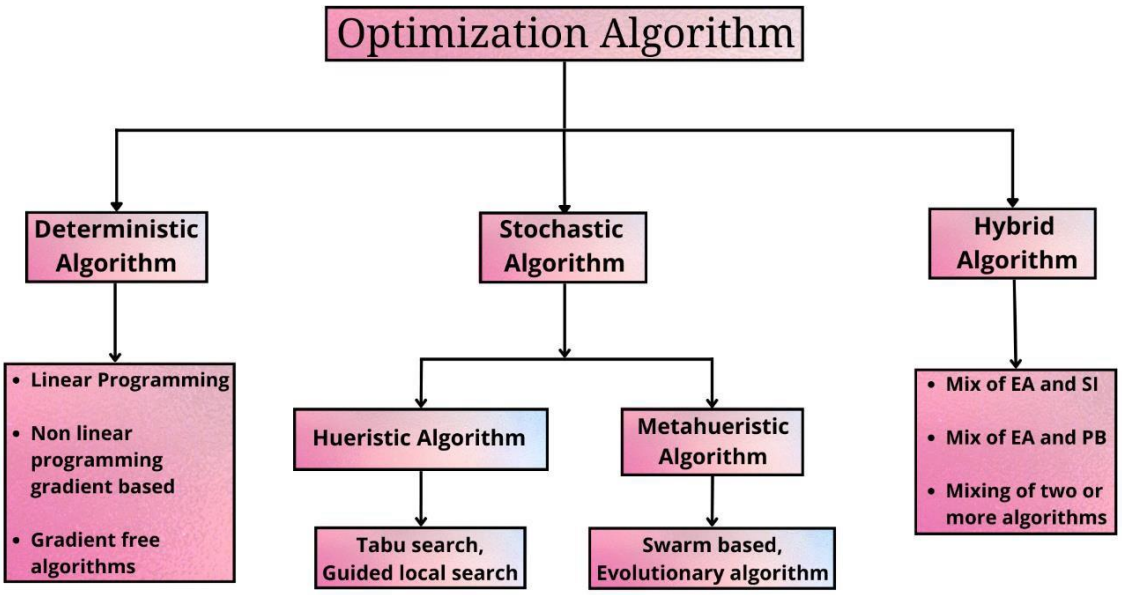


Figure 1.2: - Classification of Optimization algorithm

Conversely, the optimization process in population-based algorithms begins with a set of solutions and updates their position during optimization. GA, DE, PSO, artificial bee colony (ABC), ant colony optimization (ACO) slime mold algorithm (SMA), hunger games search (HGS), Runge Kutta optimizer (RUN), and Harris hawk's optimization (HHO), are some of the population-based algorithms. These methods have a high capacity to escape local optimal solutions because they use a set of solutions during optimization. Moreover, the exchange of information can be shared between solutions, which helps them to better search in difficult search spaces. However, these algorithms require a large number of function evaluations during optimization and high computational costs. According to the above discussion, population-based algorithms are considered more reliable and robust optimization methods than single-solution-based algorithms. Generally, an algorithm's best formulation is explored by evaluating it on different types of benchmark and engineering problems. Often, optimizers employ one or more operators to perform two phases: exploration and exploitation. An optimization algorithm requires a search mechanism to find promising areas in the search space, which is done in the exploration phase. The exploitation phase improves

the local search ability and convergence speed to achieve promising areas. The balance between these two phases is a challenging issue for any optimization algorithm. According to previous studies, no precise rule has been established to distinguish the most appropriate time to transit from exploration to exploitation due to the unexplored form of search spaces and the random nature of this type of optimizer Therefore, realizing this matter is essential to design a robust and reliable optimization algorithm.

CHAPTER 2

Literature Review

2.1 Real world application of optimization

In the 21st century, optimization algorithms have seen much use in technical contexts having to do with attaining the best possible functionality, as in" network optimization" and" search engine optimization" (Seo). Optimum and optimism (the amount or degree of something that is best or most effective, refer, respectively) derived from Latin optimum, meaning" best". Optimization theory has not only evolved in the field of computing systems but also become increasingly important and popular in various engineering applications.

Nowadays different algorithms are used in various fields like it is used to develop a version of deer hunting optimization algorithm for optimal designing of the economic aspect of the power plant whose results are then applied to two or more different analysis and after which it will be compared with some other reference algorithms to show the system efficiency. Similarly in distribution systems also, to improve the voltage profile, stability, reduction in power losses, and economic benefits, a novel nature-inspired algorithm called elephant herding optimization algorithm is used to determine the optimal distributed generation size [10].

2.2 No free lunch theorem:

No Free Lunch (NFL) [28] theorem answers the question why there is a need for more optimization techniques. This theorem logically proves that no one can proposes an algorithm for solving all optimization problems. This means that the success of an algorithm in solving a specific set of problems does not guarantee solving all optimization problems with different type and nature. In other words, all the optimization techniques perform equal in average when considering all optimization problems despite the superior performance on a subset of optimization problems. The

NFL theorem allows researchers to propose new optimization algorithms or improve/modify the current ones for solving subsets of problems in different fields, in which a simple yet effective optimization technique is proposed to optimization real problems with unknown search spaces. The NFL theorem says that all the algorithms perform equal on all optimization problems. Therefore, there are still problems that have not been solved, or they can be solved better by new algorithms. also shows that a meta-heuristic does not necessarily need an actual inspiration, and simple mathematical functions also can be used to design optimization algorithms in this field.

2.3 Sine cosine optimizer:

The SCA [29] creates multiple initial random candidate solutions and requires them to fluctuate outwards or towards the best solution using a mathematical model based on sine and cosine functions. Several random and adaptive variables also are integrated to this algorithm to emphasize exploration and exploitation of the search space in different milestones of optimization. The proposed algorithm utilizes the functions sine and cosine to explore and exploit the space between two solutions in the search space with the hope to find better solutions. The algorithm proposed in this work is completely different in terms of inspiration, mathematical formulation, and real-world application. SCA algorithm is based on sine/cosine mathematical functions to solve optimization problems.

The performance of SCA was benchmarked in test phases. A set of well-known test cases including unimodal, multi-modal, and composite functions are employed to test exploration, exploitation, local optima avoidance, and convergence of SCA. Secondly, several performance metrics (search history, trajectory, average fitness of solutions, and the best solution during optimization) were used to quantitatively and qualitatively observe and confirm the performance of SCA. The cross-section of an aircraft's wing is optimized by SCA as a real challenging case study to verify and demonstrate the performance of this algorithm in practice, The SCA algorithm obtains a smooth shape for the air foil with a very low drag, which demonstrates that this algorithm can highly be effective in solving real problems with constrained and unknown search spaces [29].

SCA algorithm is used to shape the cross-section of an aircraft's wing and is being optimized by the SCA algorithm, which demonstrate the merits of this algorithm in solving real challenging problems with a large number of constraints and unknown search spaces.

The results proved that the proposed algorithm is able to explore different regions of a search space, avoid local optima, converge towards the global optimum, and exploit promising regions of a search space during optimization effectively.

An optimization process is divided into two phases: exploration versus exploitation. In the former phase, an optimization algorithm combines the random solutions in the set of solutions abruptly with a high rate of randomness to find the promising regions of the search space. With the operators defined in SCA, the proposed algorithm theoretically is able to determine the global optimum of optimization problems due to the following reasons:

- SCA creates and improves a set of random solutions for a given problem, so it intrinsically benefits from high exploration and local optima avoidance compared to other single-solution-based algorithms.
- Different regions of the search space are explored when the sine and cosine functions return a value greater than 1 or less than -1.
- Promising regions of the search space is exploited when sine and cosine return value between -1 and 1.
- The SCA algorithm smoothly transits from exploration to exploitation using adaptive range change in the sine and cosine functions.
- The best approximation of the global optimum is stored in a variables as the destination point and never get lost during optimization.
- Since the solutions always update their positions around the best solution obtained so far, there is a tendency towards the best regions of the search spaces during optimization
- Since the algorithm considers optimization problem as black boxes, it is readily incorporable to problems in different fields subject to proper formulation of the problem.

2.4 Ant lion optimizer:

The ALO [30] algorithm mimics the hunting mechanism of antlions in nature. Five main steps of hunting prey such as the random walk of ants, building traps, entrapment of ants in traps, catching preys, and re-building traps are implemented. Antlions (doodlebugs) belong to the Myrmeleontid family and Neuropteran order (net-winged insects). The lifecycle of antlions includes two main phases: larvae and adult. A natural total lifespan can take up to 3 years, which mostly occurs in larvae (only 3–5 weeks for adulthood). Antlions undergo metamorphosis in a cocoon to become adult. They mostly hunt in larvae and the adulthood period is for reproduction r names originate from their unique hunting behaviour and their favourite prey.

An antlion larva digs a cone-shaped pit in sand by moving along a circular path and throwing out sands with its massive jaw. Several cone-shaped pits with different sizes. After digging the trap, the larvae hide underneath the bottom of the cone (as a sit-and-wait predator) and waits for insects (preferably ant) to be trapped in the pit. The edge of the cone is sharp enough for insects to fall to the bottom of the trap easily. Once the antlion realizes that a prey is in the trap, it tries to catch it. However, insects usually are not caught immediately and try to escape from the trap. In this case, antlions intelligently throw sands towards to edge of the pit to slide the prey into the bottom of the pit. When a prey is caught into the jaw, it is pulled under the soil and consumed.

After consuming the prey, antlions throw the leftovers outside the pit and amend the pit for the next hunt. Another interesting behaviour that has been observed in life style of antlions is the relevancy of the size of the trap and two things: level of hunger and shape of the moon. Antlions tend to dig out larger traps as they become hungrier and/or when the moon is full. They have been evolved and adapted this way to improve their chance of survival. It also has been discovered that an antlion does not directly observe the shape of the moon to decide about the size of the trap, but it has an internal lunar clock to make such decisions. The main inspiration of the ALO algorithm comes from the foraging behaviour of antlion’s larvae.

The ALO algorithm finds superior optimal designs for the majority of classical engineering problems employed, showing that this algorithm has merits in solving

constrained problems with diverse search spaces. The proposed ALO algorithm is able to approximate the global optimum of optimization problems due to the following reasons:

- Exploration of the search space is guaranteed by the random selection of antlions and random walks of ants around them.
- Exploitation of search space is guaranteed by the adaptive shrinking boundaries of antlions' traps.
- There is high probability of resolving local optima stagnation due to the use of random walks and the roulette wheel.
- ALO is a population-based algorithm, so local optima avoidance is intrinsically high. Intensity of ants' movement is adaptively decreased over the course of iterations, which guarantees convergence of the ALO algorithm.
- Calculating random walks for every ant and every dimension promotes diversity in the population. Antlions relocate to the position of best ants during optimization, so promising areas of search spaces are saved.
- Antlions guide ants towards promising regions of the search space.
- The best antlion in each iteration is saved and compared to the best antlion obtained so far (elite).
- The ALO algorithm has very few parameters to adjust.
- The ALO algorithm is a gradient-free algorithm and considers problem as a black box.

An algorithm should avoid all the local optima to approach and determine the global optimum. It mimics the difficulties of real search spaces by providing a massive number of local optima and different shapes for different regions of the search space. An algorithm should properly balance exploration and exploitation to approximate the global optima. For verification of the results of ALO, two well-known algorithms were employed: PSO as the best algorithm among swarm-based technique and GA as the best evolutionary algorithm.

High exploitation assists the ALO algorithm to rapidly converge towards the optimum and exploit it accurately as can also be inferred from the convergence curves.

ALO algorithm has a high level of exploration which assists it to explore the promising regions of the search space. In addition, the local optima avoidance of this algorithm in satisfactory since it is able to avoid all of the local optima and approach the global optima and the ALO shows the fastest convergence as well.

Also, ALO algorithm potentially able to solve challenging optimisation problems. Since the composite search spaces are highly similar to the natural search spaces, the ALO algorithm could solve difficult optimisation problems.

One of the behaviours of ALO is the adaptive shrinking trend of boundaries which allows more accurate exploitation of the optimum as iteration increases. It can guarantee that an algorithm eventually convergences to a point and searches locally in a search space

To prove efficiency of ALO, different problems were chosen:

- Ship propeller design using ALO
- Cantilever beam design problem
- Three-bar truss design problem
- Gear train design problem

Other conclusion remarks that can be made are as follows:

- Random selection of antlions using a roulette wheel guarantees exploration of the search space.
- Random walks of ants around the antlions also emphasize exploration of the search space around the antlions.
- The use of random walk and roulette wheel assist the ALO algorithm to resolve local optima stagnations.
- Local optima avoidance is very high since the ALO algorithm employs a population of search agents to approximate the global optimum.

2.5 Artificial ecosystem-based optimizer:

It is a novel nature-inspired meta-heuristic optimization algorithm, named artificial ecosystem-based optimization (AEO) [31]. AEO is a population-based optimizer motivated from the flow of energy in an ecosystem on the earth, and this algorithm mimics three unique behaviours of living organisms, including production, consumption, and decomposition. The overall comparisons suggest that the optimization performance of AEO outperforms that of other state-of the-art counterparts. Especially for real-world engineering problems, AEO is more competitive than other reported methods in terms of both convergence rate and computational efforts

'Ecosystem,' an intuitively appealing concept. Owing to this appeal, the ecosystem concept has become increasingly important and popular. An ecosystem is the complex of living organisms, their physical environment, and all their interrelationships in a particular unit of space. An ecosystem can be classified into abiotic and biotic constituents. Abiotic constituents include sunlight, water, air, and other non-living elements. Biotic constituents include all living organisms. The flow of energy and the cycling of nutrients are chief driving forces maintaining the normal ecological order in an ecosystem, where all living organisms play a significant role. Living organisms can be categorized into producer, consumer, and decomposer. Producers do not need to obtain energy from other organisms, and most producers are any kind of green plant that gets their food energy directly through the process of photosynthesis. During photosynthesis, carbon dioxide and water react together in the presence of sunlight to produce a sugar called glucose and oxygen. Plants use this sugar to make many things, such as leaves, fruits, wood, and roots. Producers generally provide essential food for herbivore consumers and omnivore consumers. Consumers are animals that cannot make their food; thus, they must feed on producers or other consumers to obtain energy and nutrients. Consumers can be categorized into carnivores, herbivores, and omnivores. Animals that eat only producers (plants) are called herbivores. Animals that eat both producers and other animals are called omnivores. Animals that eat only other animals are called carnivores. Decomposers are an organism that feeds on both dead

plants (producers) and animals (consumers) or on the waste from living organisms. Decomposers include most bacteria and fungi. When an organism dies, decomposers break down the remains and convert them into simple molecules, such as carbon dioxide, water, and minerals. Then, these energy forms may be absorbed by producers to again make sugar and oxygen through photosynthesis, starting the cycle all over again. Producers, consumers, and decomposers interacting with one another compose a food chain. A food chain describes who feeds whom in an ecosystem, where every life form needs to get food energy to achieve the movement of nutrients. A food chain is a possible connected path that energy and nutrients can follow through an ecosystem, and it shows the different levels of eating within an ecosystem. Many overlapped food chains form a food web that depicts how food chains are interconnected. A food web is very complex because most consumers feed on producers or/and different kinds of consumers and meanwhile probably be fed on other different kinds of consumers. Producers are generally at the beginning of feed webs and most food chains. Consumers are the most complex of three types of organisms. The blue arrows represent the energy transfer pathway. Typically, the ecosystem develops this mechanism of energy transfer as a strategy to maintain the stability of species, and it is able to keep the ecological balance over a long term. Therefore, there are reasons to believe that this mechanism of energy transfer can shape and sustain a sound and stable ecosystem.

Artificial ecosystem-based optimization algorithm employs three operators, including production, consumption, and decomposition. The first operator is mainly to enhance the balance between exploration and exploitation. The second operator is used to improve the exploration of the algorithm. For the third operator, it is proposed to promote the exploitation of the algorithm. AEO generally follows the following several rules to search for a solution.

- The ecosystem as a population includes three kinds of organisms: producer, consumer, and decomposer.
- There is only one producer as an individual in a population.
- There is only one decomposer as an individual in a population.

- The other individuals of a population are consumers, each of which is chosen as a carnivore, an herbivore, or an omnivore with the same probability.
- The energy level of each individual in a population is evaluated by its function fitness value. The population is sorted in the descending order of function fitness value, so the higher function fitness value indicates the higher energy level for a minimization problem.

In an ecosystem, the producer may generate food energy with carbon dioxide, water, and sunlight, as well as the nutrition provided by the decomposer. Likewise, in AEO, the producer (the worst individual) in a population needs to be updated by the low and upper limits of search space and the decomposer (the best individual), and this updated individual will guide other individuals including herbivore and omnivore in the population to search for different regions. After the producer accomplishes production operator, all the consumers may perform consumption operator. To obtain food energy, each consumer may eat either a randomly chosen consumer with the lower level of energy or a producer, or both.

Levy flight, as a mathematic operator mimics food searching of many animals like cuckoo, bumblebees, deer, and lion. Levy flight is a random walk which can efficiently explore the search space as the length of some steps is much longer in the long run, showing that it is promising in finding the global optimum. Therefore, Levy flight was often added to nature-inspired algorithms to enhance their optimization performance. However, there are two drawbacks to this behaviour, including complexity and several parameters to be tuned. At a point, a simple, parameter-free random walk with the feature of Levy flight, called consumption factor, is proposed. Decomposition is a very vital process in terms of the functioning of an ecosystem, and it provides essential nutrients for the growth of the producer. During the decomposition, when each individual in the population dies, the decomposer will decay or break down chemically its remains.

With the formulation of AEO algorithm and the observation of its optimization performance, the following remarks are made.

- AEO is motivated from three energy transfer mechanism in an ecosystem, including production, consumption, and decomposition, which are effective in improving the optimizing performance of AEO from different aspects.
- The production assists AEO to produce a candidate solution drifting from a randomly generated position to the best position with the increase in iterations, and this solution will guide other individuals to perform consumption operator during the consuming process. This behaviour contributes greatly to the balance between the explorative and exploitative search.
- The consumption allows AEO to update the solutions of individuals with respect to either the solution offered by the production process or the solution of the randomly chosen individual with higher energy level, or both. It can improve the exploration of AEO.
- The consumption factor encourages AEO to perform a global search.
- In consumption process, each consumer is randomly chosen as a carnivore, an herbivore, or an omnivore with the same probability. An herbivore updates its solution based on the solution offered by the production process. A carnivore updates its solution based on the randomly chosen individual with the higher energy level. An omnivore updates its solution based on between both the solution offered by the production process and the randomly chosen solution with higher energy level.
- The decomposition enables AEO to update the solutions of individuals based on the best solution in the population via three key coefficients. It can enhance the exploration of AEO.
- AEO is very simple to implement and does not require any other parameter to be adjusted besides both the number of population and the maximal iteration.

AEO has high local optima avoidance resulting from an appropriate balance between exploration and exploitation. Such capability originates from the fact the production operator guides individuals to search from a randomly generated region far away from the best individual to a region close to the best individual as the iterations increase.

Wilcoxon signed-rank test (WSRT) can effectively assess the overall performance of algorithms. WSRT is a nonparametric test which can be used to check for statistical significance difference between two algorithms. To determine whether the one outperforms the other one with certain statistical significance. AEO shows a better performance at 95% significance level for the majority of functions.

AEO algorithm has two control parameters: the maximal number of iterations and the size of population. The sensitivity of the control parameters is investigated by changing each value of both parameters while keeping the other one unchanged. The size of population: To explore the effect of the population size on the performance of AEO. The results obtained by AEO are able to offer very high precision for varying population sizes. Varying the population size does not result in a significant change in precision. AEO obtains better convergence toward the global optimum. AEO can converge toward the global optimum as the number of iterations increases. AEO provides good convergence rate for different maximal number of iterations.

Many optimization problems in the real world include multiple decision variables. It is very necessary for us to investigate the effect of scalability on the problems using AEO algorithm. The results demonstrate that increasing dimensionality of decision variables has the least effect on the performance of AEO.

Eight constrained engineering design problems, including welded beam design, three-bar truss design, tension/compression spring design, cantilever beam design, pressure vessel design, speed reducer design, rolling element bearing design, and multiple disk clutch brake design, were utilized to the validity of this proposed algorithm. However, these problems are all constraint optimization problems that involve inequality and equality constraints. For simplicity, penalty functions are used to handle these constraint problems in AEO. When using penalty functions, every search individual who violates any constraint with any level is assigned a big function fitness value. In this way, these constrained problems can be converted into unconstrained problems by introducing penalty functions. This method is very effective and is easy to implement for AEO without the need for modification of the algorithm. AEO does not need to adjust any other parameter except for the size of population and the maximal number

of function evaluations, indicating the consistence of AEO performance among different problems.

AEO is more promising to tackle real-world problems requiring a faster convergence rate and less computational expense with a specified precision for final solutions. AEO provides enough competitive results compared to other considered meta-heuristic methods. However, the computational efficiency and solution quality depend largely on presentation and complexity of problems. This is a general characteristic not just for AEO, but for most metaheuristics. Therefore, algorithm tends to handle the real-world problems focusing on convergence efficiency and computational expense with an acceptable precision for final solutions. For future work, the binary AEO may be presented to deal with complex discrete problems. AEO could also be tailored to address multi-objective optimization.

2.6 Grey wolf optimizer:

A meta-heuristic called Grey Wolf Optimizer (GWO) [32]is inspired by grey wolves (Canis lupus). The GWO algorithm mimics the leadership hierarchy and hunting mechanism of grey wolves in nature. Four types of grey wolves such as alpha, beta, delta, and omega are employed for simulating the leadership hierarchy. In addition, the three main steps of hunting, searching for prey, encircling prey, and attacking prey, are implemented. The results on benchmark function show that the GWO algorithm is able to provide very competitive results compared to these well-known meta-heuristics. The three classical engineering design problems (tension/compression spring, welded beam, and pressure vessel designs) and a real application of the proposed method in the field of optical engineering was studied using GWO. The results of the classical engineering design problems and real application always proves that the proposed algorithm is applicable to challenging problems with unknown search spaces.

Grey wolf (Canis lupus) belongs to Canidae family. Grey wolves are considered as apex predators, meaning that they are at the top of the food chain. Grey wolves mostly prefer to live in a pack. Of particular interest is that they have a very strict social dominant

hierarchy. The leaders are a male and a female, called alphas. The alpha is mostly responsible for making decisions about hunting, sleeping place, time to wake, and so on. The alpha's decisions are dictated to the pack. However, some kind of democratic behaviour has also been observed, in which an alpha follows the other wolves in the pack. In gatherings, the entire pack acknowledges the alpha by holding their tails down. The alpha wolf is also called the dominant wolf since his/her orders should be followed by the pack. The alpha wolves are only allowed to mate in the pack. Interestingly, the alpha is not necessarily the strongest member of the pack but the best in terms of managing the pack. This shows that the organization and discipline of a pack is much more important than its strength. The second level in the hierarchy of grey wolves is beta. The betas are subordinate wolves that help the alpha in decision-making or other pack activities. The beta wolf can be either male or female, and he/she is probably the best candidate to be the alpha in case one of the alpha wolves passes away or becomes very old. The beta wolf should respect the alpha, but commands the other lower-level wolves as well. It plays the role of an advisor to the alpha and discipliner for the pack. The beta reinforces the alpha's commands throughout the pack and gives feedback to the alpha. The lowest ranking grey wolf is omega. The omega plays the role of scapegoat. Omega wolves always have to submit to all the other dominant wolves. They are the last wolves that are allowed to eat. It may seem the omega is not an important individual in the pack, but it has been observed that the whole pack face internal fighting and problems in case of losing the omega. This is due to the venting of violence and frustration of all wolves by the omega(s). This assist satisfying the entire pack and maintaining the dominance structure. In some cases, the omega is also the babysitters in the pack. If a wolf is not an alpha, beta, or omega, he/she is called subordinate (or delta in some references). Delta wolves have to submit to alphas and betas, but they dominate the omega. Scouts, sentinels, elders, hunters, and caretakers belong to this category. Scouts are responsible for watching the boundaries of the territory and warning the pack in case of any danger. Sentinels protect and guarantee the safety of the pack. Elders are the experienced wolves who used to be alpha or beta. Hunters help the alphas and betas when hunting prey and providing food for the pack. Finally, the

caretakers are responsible for caring for the weak, ill, and wounded wolves in the pack. In addition to the social hierarchy of wolves, group hunting is another interesting social behaviour of grey wolves. The main phases of grey wolf hunting are as follows: Tracking, chasing, and approaching the prey. Pursuing, encircling, and harassing the prey until it stops moving. Attack towards the prey.

2.7 Arithmetic optimizer

This work proposes a new meta-heuristic method called Arithmetic Optimization Algorithm (AOA) [33] that utilizes the distribution behaviour of the main arithmetic operators in mathematics including (Multiplication (M), Division (D), Subtraction (S), and Addition (A)). AOA is mathematically modelled and implemented to perform the optimization processes in a wide range of search spaces. The performance of AOA is checked and several real-world engineering design problems to showcase its applicability. The analysis of performance, convergence behaviours, and the computational complexity of the proposed AOA have been evaluated by different scenarios. Experimental results [33] show that the AOA provides very promising results in solving challenging optimization problems compared with eleven other well-known optimization algorithms.

Since population-based algorithms seek to find the optimal solution of optimization problems stochastically, getting a solution in a single run is not guaranteed. Nevertheless, the probability of getting the global optimal solution, for the given problem, is increased by a sufficient number of random solutions and optimization iterations. Despite the differences between meta-heuristic algorithms in the area of population-based optimization methods, the optimization process consists of two main phases: exploration versus exploitation. The former refers to extensive coverage of search space using search agents of an algorithm to avoid local solutions. The latter is the improvement accuracy of obtained solutions during the exploration phase.

Arithmetic is a fundamental component of number theory, and it is one of the important parts of modern mathematics, along with geometry, algebra, and analysis. Arithmetic operators (i.e., Multiplication, Division, Subtraction, and Addition) are the traditional

calculation measures used usually to study the numbers. We use these simple operators as a mathematical optimization to determine the best element subjected to specific criteria from some set of candidate alternatives (solutions). Optimization problems occur in all quantitative disciplines from engineering, economics, and computer sciences to operations research and industry, and the improvement of solution techniques has attracted the interest of mathematics for eras. The main inspiration of the proposed AOA arises from the use of Arithmetic operators in solving the Arithmetic problems. In the following subsections, the behaviour of Arithmetic operators (i.e., Multiplication, Division, Subtraction, and Addition) and their influence in the proposed algorithm will be discussed. The hierarchy of Arithmetic operators and its dominance from the outside to the inside. AOA is then proposed based on the mathematical model. The average running time of the proposed AOA algorithm compared to other well-known optimization algorithms. It can be observed that the proposed AOA needs less running time than other comparative algorithms. Since AOA is a population-based algorithm, there is no need for optimization processes, i.e., Multiplication, Division, Subtraction, and Addition. Consequently, we concluded that the computational performance of the proposed AOA algorithm is sufficiently better than the other comparative algorithms. The AOA is proficient in achieving high-quality solutions and in overwhelming other competitors.

The behaviour of Arithmetic operators in mathematical calculations, a novel meta-heuristic optimization algorithm, the Arithmetic Optimization Algorithm (AOA), is proposed. Counter to most of the well-known optimization algorithms, AOA has an easy and straightforward implementation, according to its mathematical presentation, to adapt to tackle new optimization problems. It does not need to adjust many parameters except the population size and stopping criterion, which are standard parameters in all optimization algorithms. The random and adaptive parameters also expedited the divergence and convergence of the search solutions in the AOA.

Finally, the proposed AOA is was used to solve five real-life engineering design problems (welded beam design problem, tension/compression spring design problem, pressure vessel design problem, 3-bar truss design problem, and speed reducer design

problem) to test and confirm its performance. According to the obtained results, the proposed AOA can find better solutions for most of the examined problems compared to other well-known optimization algorithms in regard to solutions' quality and computational performance. Moreover, the results of the proposed AOA also sufficiently prove its superiority on the ability to avert trapping of the local optima.

2.8 Application in power system

The outdated old grid is proving to be a hindrance in the vision of the smart home and this old grid system is needed to be upgraded for future support. This system of grid consists of 3 main components. The components are:

• Smart systems for infrastructure

• Intelligent energy management subsystem

• system based on smart protection.

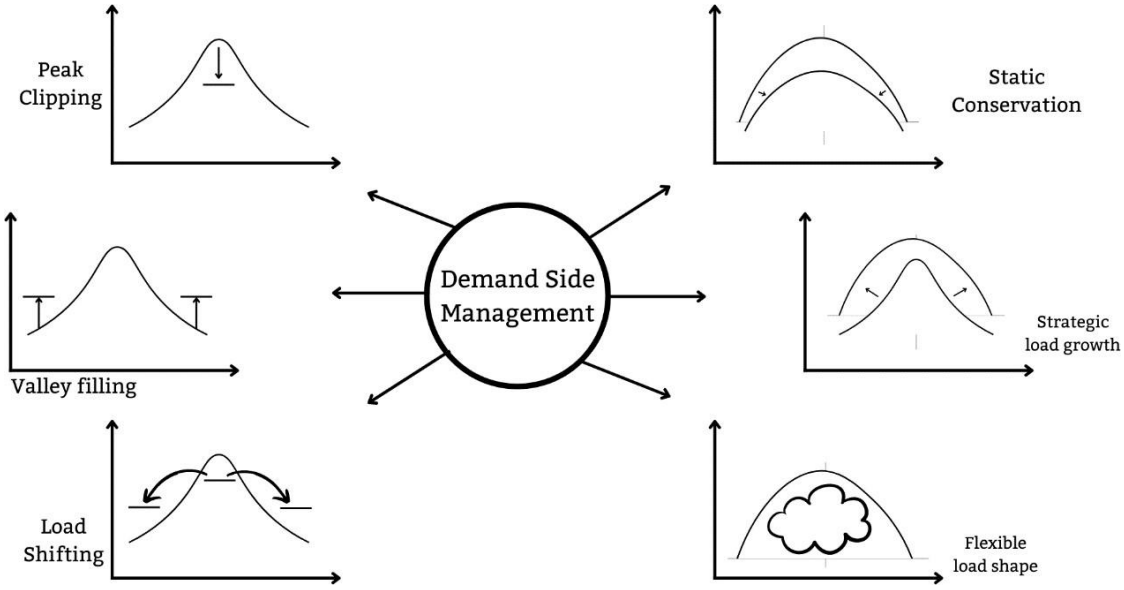


Fig 2.1: - Demand Side Management

The smart management system's essential feature is demand-side management. Demand-side management can be overviewed as the management of demand and load to get the minimum generation cost, which can be achieved using load schedule. The experiment conducted by Ahmed Shaban and Hagag Maher and Mahmoud

Elbayoumi and Suzan Abdelhady [11] demonstrate and results validate the performance of the Cuckoo scheduling approach and demonstrate its usefulness for power system applications. The test of load scheduling was carried out by researchers in an academic building in Egypt where they tried to minimize the utilization of PV output while minimizing the energy cost. The required power was reduced significantly from the grid. The cost reduction by employing load scheduling using the Cuckoo scheduling approach was on average 57% to 80%, and the peak load was reduced by 54%. A genetic algorithm based on natural selection was employed in research [12] to see the impact of demand-side management load shifting for both utilities and customers.

The experiment was carried out in a residential area. Two variants of the genetic algorithm were modelled with different objectives. The first model was maximizing the load, while the second was minimizing the energy cost. The result showed a decrease in total system peak demand. The system was able to maximize the overall system load factor. The maximization of load factor led to a reduction in the electricity bill.

2.9 Application in wireless sensor network

In literature, many algorithms are proposed to prolong the lifetime of the network based on data aggregation. Some references are concentrating on establishing energy efficient topologies to save energy.

In [13], an algorithm named PEDAP (Power Efficient Data gathering and Aggregation Protocol) is proposed, which generates Minimum Spanning Tree (MST) by utilizing the Prim algorithm. It is a near-optimal minimum spanning tree-based routing scheme. But it cannot guarantee the energy efficiency of the network, then an energy-aware version of PEDAP, namely PEDAP-PA is proposed to maximize the network lifetime. However, the sink node needs to periodically broadcast and calculate the MST of the network, thereby, the workload of the sink node is high.

In [14], a Local Minimum Spanning Tree algorithm called LMST is presented to establish the network topology, although it can effectively reduce the average degree of nodes, some prominent problems still emerge. Each node needs to periodically calculate and update the MST (Minimum Spanning Tree) locally which leads to high

computational overhead for each node. Moreover, each node needs to communicate with its neighbours to obtain the energy condition of neighbours, which still costs a lot of energy.

In [15], a localized, self-organizing, robust, and energy efficient data aggregation algorithm named L-PEDAP is proposed which combines LMST with RNG [16]. Although it is proven to have the capability of prolonging the network lifetime, its topology construction procedure is nearly identical to that of LMST, hence, it cannot be considered an energy-efficient algorithm. To reduce the cost of constructing and maintaining the energy-efficient topology, a heuristic method called ACA (Ant Colony Algorithm) which is based on the ant colony algorithm is proposed in [17]. In the ant colony optimization, a colony of artificial ants is used to construct solutions guided by the pheromone's trails and heuristic information [18].

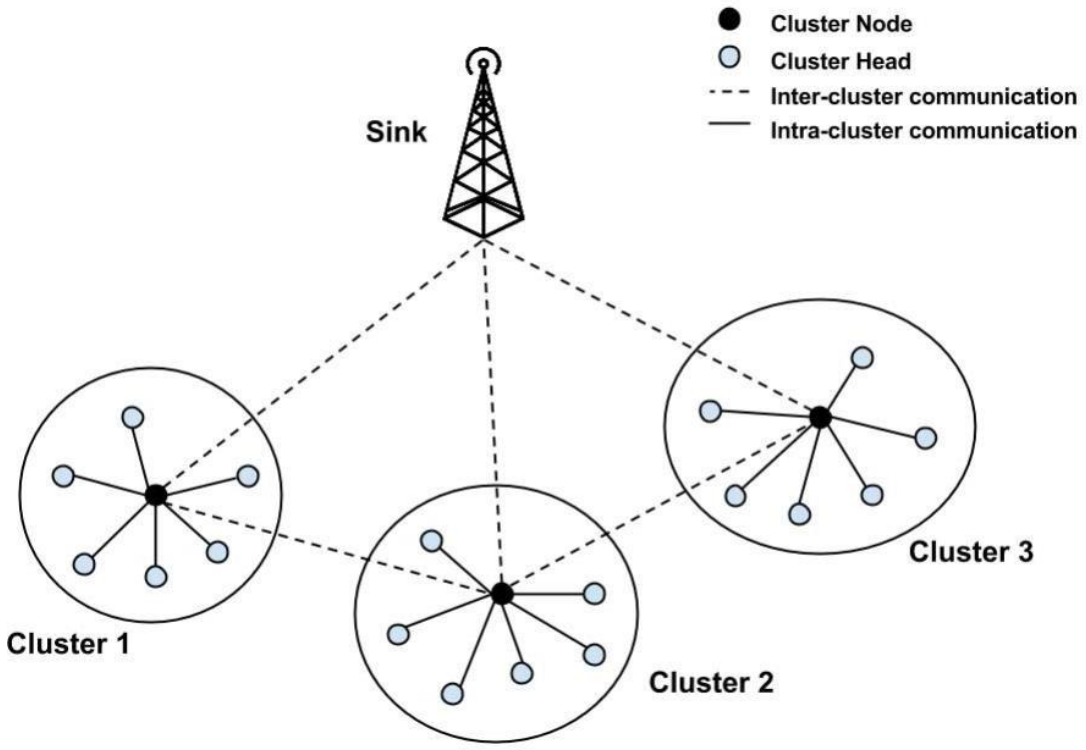


Fig 2.2: - Clustered WSNs Topology

This behaviour enables ants to find the shortest paths between the food source and the nest in a random manner. The author initiatively designs the rules for depositing and evaporating pheromones. However, the requirements of depositing pheromones can be easily met. When depositing pheromones, each node needs to communicate with its neighbours. Therefore, there are lots of communications within the pheromones adjusting period, which leads to much energy cost overhead. Therefore, it cannot be considered an energy-efficient algorithm either. Moreover, although some other methods do not depend on energy efficient topologies to save energy, some literatures are still valuable to be mentioned. In [30], the authors describe the design and implementation of a running system for energy-efficient surveillance which allows a group of the cooperating sensor to detect and track the positions of
moving vehicles. That method can trade off the energy-awareness and surveillance performance by adjusting the sensitivity of the system. In [19], data aggregation is motivated by optimizations of aggregation queries. The authors present a variety of techniques to improve the reliability and performance of the proposed solutions. In [20], the authors explore two methods to further reduce energy consumption in the context of network aggregation in sensor networks. Firstly, a group-aware network configuration method is designed, which groups the sensors into clusters. Secondly, a framework to use temporal coherency tolerances in conjunction with aggregation to save energy is proposed**.**

CHAPTER 3

Design and implementation

3.1 Introduction: Wireless sensor network model

As the Internet of Things (IoT) becomes real, several problems, inherent to energy-constrained nodes and mesh networks, arise. One of these issues is related to the optimal transmission power for each node of the network, for which the whole network is connected but no energy is wasted. Even though research in this field has been carried out for more than a decade, as can be seen in [21], it is still of concern as research is still found to be being developed in this field, as shown in the survey by Aziz et al. [22]. New challenges appear with the ever-increasing heterogeneity of such networks to form the IoT, requiring new analysis of previous solutions and also whole new solutions.

Although a centralized algorithm may present an efficiency handicap in distributed scenarios like IoT due to increased communication overhead with coordinator nodes, the centralized optimal solution still needs to be known, even if only theoretically, to measure how close to the optimal solution a proposal result can be. Therefore, several studies in the literature apply centralized solutions to wireless sensor networks (WSNs). One of the methods used is particle swarm optimization (PSO) and some of these proposals have been gathered in a survey by Kulkarni and Venayagamoorthy [23].

According to the previously mentioned work, PSO is easy to implement, computationally efficient, and fast to converge. Thus, it will be considered in this work. The set of issues addressed by centralized solutions for WSNs includes optimal node deployment, node localization, energy-aware clustering, and data aggregation [3]. In this work, communications inside a single cluster will be studied.

Considering node's connectivity and deployment, a multiple nature-based algorithms is used to find a quasi-optimal transmission power for each node to minimize energy

consumption while keeping the cluster fully connected. Then, they will be compared to each other to assess the energy-saving gain. In the present research phase, real technologies have different bandwidths, power consumption, and range, e.g., WIFI, ZigBee, and Bluetooth [24], have not been considered, although the frequency parameter can be modified to match each technology.

3.2 Energy saving in the network:

The main limitation of wireless sensors is still their constrained energy capacity, thus keeping research active in this field [25]. Heinzelman et al. [26] have begun researching energy efficiency in WSNs more than a decade ago. They have proposed the use of clustering and beamforming to prevent unnecessary transmissions. Since then, sensor nodes have evolved, acquiring new capabilities. Hence, notwithstanding the maturity of the research field, this evolution that leads to the IoT will still require research to be done.

The first step to propose methods to save energy is developing sensor energy models. According to Abdelaal and Theel [25], three subsystems draw energy from the batteries - the sensing unit, the processor, and the communication unit. Moreover, the authors say that several wireless node events can be adjusted to mitigate wasting power. Namely, they are packet overhead, idle listening, overhearing, over-emitting, collisions, and state transitions. The goal of the present introductory research is to avoid state transitions by finding the optimal transmission power for each node in the network, to fully connect sensor nodes in a single cluster. For now, the other energy-wasting events will not be considered.

Although changing states can waste some of the sensor energy, it may be a mechanism to save energy. For instance, switching to a sleep state, where most of the node's capabilities are disabled, results in energy savings, as shown by Mendes et al. [27]. In this research phase, this work is not going to consider different node states because it will focus on network connectivity. For stationary sensors, once the network

connectivity has been achieved, the network objective can switch to changing modes in its next phase.

3.3 Modelling

The optimal solution to find the minimum necessary power to connect every node of a wireless sensor network would be to calculate the minimal power for each sensor node to reach only its closest neighbour if it is placed on the edge of the network, or a set of nodes to connect the edges if it is an intermediate node. However, as the number of nodes in the cluster increase, checking for the closest nodes and keeping the whole network connection would lead to high computational overhead. Thus, a suboptimal, but the less computing-intensive solution could be applied.

3.4 Neighbours matrix

The scenarios considered in this work consist of a single cluster with N wireless sensors. Their objective is to send their measurement packets to a sink node through the mesh network they will create. The sensors and sink positions will be bounded to an L-sided square. Furthermore, the global neighbour' matrix $\Theta_{(y)}$ is calculated as a function of the transmitted power vector $\boldsymbol{\gamma}$ according to

$$\Theta_{(y)} = \begin{cases} 0 & if \rho_j < \rho_{th} \\ 1 & if \rho_j > \rho_{th} \end{cases}$$

where $\boldsymbol{\rho_j}$ is the received power at node j, when i transmit with power $\boldsymbol{\Upsilon_i}$, and $\boldsymbol{\rho_{th}}$ the receiver sensitivity. Thus, two nodes are connected when the first transmit with enough power such that the received signal has more power than the receiver sensitivity. As shown in Figure 2.1, the circles represent the distance from the node in its centre where the measured power is exactly $\boldsymbol{\rho_{th}}$. This means that another node inside the circle can receive its signal.

$$\frac{P_r}{P_t} = \frac{A_r A_t}{d^2 \lambda^2}$$

Furthermore, the received power is a function of the physical distance between the nodes and the transmitted power, related by the Friis formula.

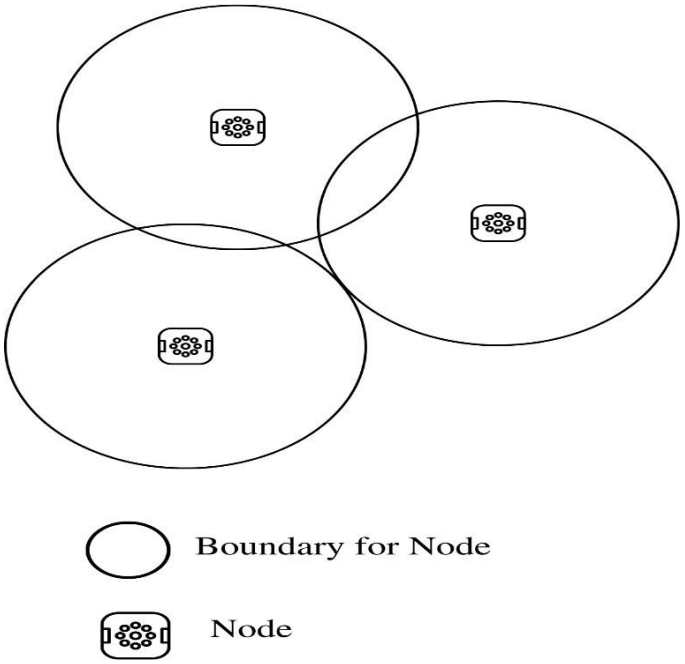


Fig: - 3.1 System Model

where $\mathbf{P_r}$ is the received power, $\mathbf{P_t}$ is the transmitted power, $\mathbf{A_r}$ and $\mathbf{A_t}$ are the effective areas of the receiving and transmitting antennas, respectively, d is the distance between the antennas and ٦ is the wavelength, given by the speed of light c divided by the signal frequency f. Since this work does not focus on a specific hardware platform for the sensor nodes, the simplest antenna model will be chosen to model the effective antenna areas. Thus, considering a single transmission and reception isotropic antenna at each sensor node, the effective areas are given by

$$A_{isotropic} = \frac{\lambda^2}{4 \times \Pi}$$

and, thus, making $\mathbf{P_R} = \mathbf{P_t} = \mathbf{A_{isotropic}}$, the power ratio simplifies to

$$\frac{P_r}{P_t} = \left(\frac{\lambda}{4 \times \Pi \times d}\right)^2$$

After computing the neighbour's matrix, an algorithm is used to determine if the network is connected, which is explained next.

3.5 Determining if the Network is Fully Connected

A fully-connected condition is achieved when there is at least one connection for each node, and it needs to compose a path involving all nodes, as seen in Figure 3 (a).

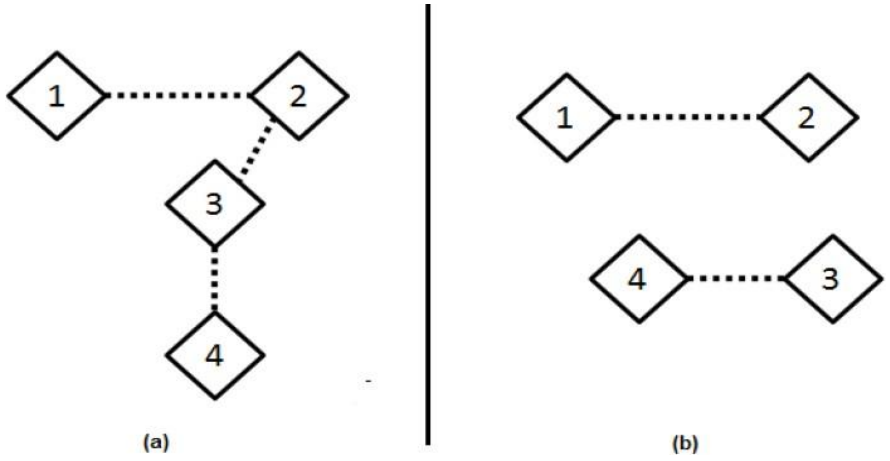


Fig 3.2: - (a) Network fully connected. (b) Disconnected network.

The first step for the connectivity determination is to calculate the Laplacian Matrix of the neighbour matrix, being $\mathbf{n_i}$ the $\mathbf{I^{th}}$ node, and $\mathbf{n}$ the total amount of nodes.

$$L = (l_{ij})_{n \times n}$$

$$l_{ij} = \begin{cases} deg(n_i) & if i = j \\ -1 & if i \neq j and \Gamma_{ij} = 1 \\ 0 & otherwise \end{cases}$$

where deg($\mathbf{n_i}$) represents the number of nodes connected to node $\mathbf{n_i}$, and such value is calculated according to;

$$deg(n_i) = \Gamma_{ij}^2 \leftrightarrow i = j$$

That means the number of connections to $\mathbf{n_i}$ is the same term of the neighbor matrix squared, when $\mathbf{i = j}$. Once the Laplacian Matrix has been calculated, the second step is to find the eigenvalues $\boldsymbol{\psi}$ of the Laplacian matrix $\mathbf{L}$, through equation

$$\mathrm{L.E} = \omega.\mathrm{E}$$

where E is a column vector with length n, called eigenvector, which must satisfy the equality and match each possible Laplacian eigenvalue ω. The values of μ that satisfy are the eigenvalues associated to each eigenvector, and could be grouped inside a vector μ ,

$$\mu = [\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \dots., \omega_n]^t$$

with $\omega_1 < \omega_2 < \omega_3 < \omega_4 < \omega_5 < \cdots . < \omega_n$ This means that the second smallest Laplacian eigenvalue ω_2 , also called algebraic connectivity of the Neighbour Matrix, and the fully-connected condition is finally achieved if, and only if, ω_2 is positive.

In conclusion, determining if a network is fully connected demands observing at least one connection per node, and the second smallest Laplacian eigenvalue of the neighbour matrix must be positive. Once these conditions are met, the transmission power of each node is enough to establish a fully-connected cluster.

3.6 Test cases

TABLE 3.1: Coordinates of nodes

Case 1		Case 2		Case 3		Case 4		Case 5		Case 6		Case 7		Case 8		Case 9		Case 10	
X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y
-3	-10	6	-2	-3	7	3	-3	6	-2	3	-3	-9	-5	-10	5	-9	-5	-10	1
6	-9	-5	-2	2	1	7	1	-5	-2	7	1	6	-10	0	0	6	-10	8	4
6	-7	-8	-8	9	-4	-3	9	-8	-8	-3	9	9	5	8	2	9	5	-7	-3
7	1	9	10	5	5	8	1	9	10	8	1	0	2	2	8	0	2	-1	10
1	2	2	-9	-3	1	3	2	2	-9	3	2	-6	-1	6	2	-6	-1	-7	7
-9	-3	-6	-3	-9	-9	-6	-4	-6	-3	-6	-4	10	1	-7	-5	10	1	3	-3
1	-9	7	-10	1	6	-1	-6	7	-10	-1	-6	0	-6	8	-10	0	-6	-6	-2
1	5	-10	-7	9	-8	7	-6	-10	-7	7	-6	0	3	0	-7	0	3	0	-8
1	9	3	5	1	-1	-6	-7	3	5	-6	-7	4	-2	10	4	4	-2	2	-6
-4	9	3	-1	-10	-3	-6	-1	3	-1	-6	-1	-3	10	0	-1	-3	10	-2	2

There are ten different cases design to test all the optimization algorithm. Table 2.1 present the coordinates of sensors of all the 10 cases. The importance of coordinates of sensors can't be overlooked. Random placement of the sensor nodes is usually preferred in a large scale WSN because it is easy and less expensive. However, it can cause holes formulation and cannot guarantee to achieve the required targets. Therefore, it is necessary to optimize the node placement of the WSN. The node placement problem is one of the fundamental problems when building wireless sensor networks. The solutions to node placement problem include node placement strategy and node placement optimization.

The node placement strategy is generally used for the deterministic deployment of initial nodes, which can obtain the appropriate node placement through a certain strategy at the beginning of the WSN. The node placement optimization is generally used to optimize the location of existing nodes. If mobile nodes (such as flying nodes or unmanned aerial vehicles) exist, the position of the nodes can be adjusted whenever needed. In some harsh environments, sensor nodes can only be randomly placed. Then a node placement that meets the application requirements should be obtained by the node placement optimization algorithm, and the nodes will move to the appropriate positions. This work assume optimal placement of nodes.

CHAPTER 4

Simulation and Results

4.1 Importance of benchmarking:

Comparing, or benchmarking, of optimization algorithms is a complicated task that involves many subtle considerations to yield a fair and unbiased evaluation. In the most general sense, benchmarking is the comparison of one or more products to an industrial standard product over a series of performance metrics. In the case of benchmarking optimization algorithms, the products are the specific implementations of given algorithms, and the performance metrics are generated by running the implementations on a series of test problems. If one algorithm runs faster, uses less memory, and returns a better final function value, on all possible problems, then it can be considered better than the alternative. Of course, in practice such a clear conclusion seldom arises. Thus, interpreting the conclusions of algorithmic comparisons can be tricky.

Benchmarking optimization algorithms can have great practical value. It can reveal both strengths and weaknesses of an algorithm, which allows for better research focus. It can aid in determining if a new version of optimization software is performing up to expectations. And, it can help guide end-users in selecting a good choice of algorithm for a particular real-world problem. However, when done poorly, benchmarking optimization algorithms can also be misleading. It can hide algorithm's weaknesses (or strengths), report improvements that do not exist, or suggest the incorrect algorithmic choice for a given situation. In optimization benchmarking many subjective choices are made, such as the test set to solve, the computing environment to use, the performance criteria to measure, etc. The parameters considered to evaluate the results are: -

1) Magnitude analysis
    a) Simplistic Method
2) Box plot analysis
3) Convergence analysis

4.2 Magnitude analysis:

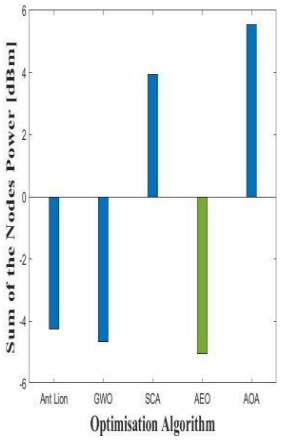
4.2.1 Simplistic method:

When no power adaptation system is used by the WSN, the nodes transmitted power will not change throughout the network operation. Thus, this method will sweep a defined transmission power range and the full-connection of the network will be checked. Bounded by the minimum transmission power, Y_{min} and the maximum one, Y_{max}, the algorithm will set, initially, all nodes transmission powers. Then, the previously presented method will be applied to calculate the global neighbour's matrix and the full connection of the network can be verified. If the network is determined to be connected, all nodes transmission powers are decreased by Δy, the neighbour's matrix is recalculated, and the network connectivity is checked again. In this direction, the algorithm stops at the last transmission power that was able to connect the whole network. Otherwise, if the first transmission power resulted in a disconnected network, all nodes transmission powers are increased by Δy Then, the same method is applied, stopping at the first transmission power capable of connecting the whole network.

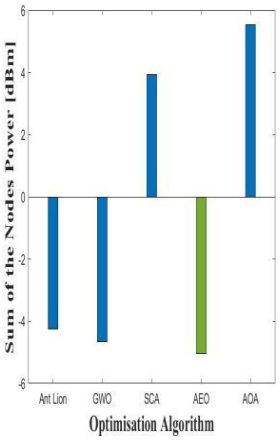
4.2.2 Result analysis:

TABLE 4.1: Optimal cluster transmission power by the SM and 5 optimizers.

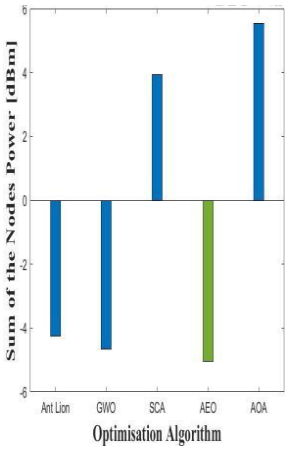
Case No.	Ant-lion	GWO	SCA	AEO	AOA	SM
1	-4.253	-4.655	3.943	-5.043	5.538	2.808
2	-6.966	-7.307	2.414	-7.680	5.039	-2.76
3	-4.329	-4.651	1.276	-5.042	5.380	2.791
4	-5.568	-5.894	3.265	-6.258	3.393	3.851
5	-6.914	-7.253	2.725	-7.680	4.364	-2.786
6	-5.493	-5.938	2.288	-6.258	1.590	3.867
7	-4.386	-4.655	3.613	-5.043	6.898	-1.329
8	-6.621	-6.971	0.750	-7.287	5.155	-2.753
9	-4.627	-4.788	4.135	-5.042	3.815	-1.313
10	-3.838	-4.232	3.083	-4.505	7.231	1.235



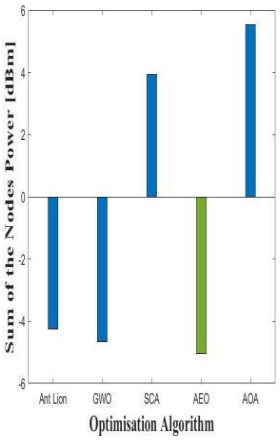
(a) Case no. 1



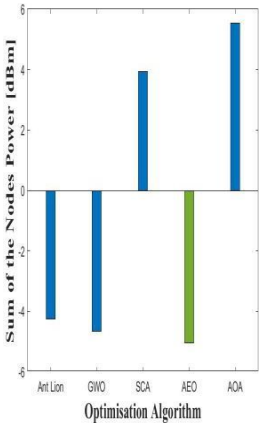
(b) Case no. 2



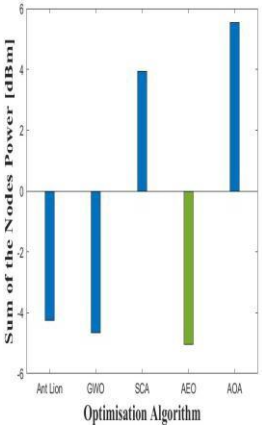
(c) Case no. 3



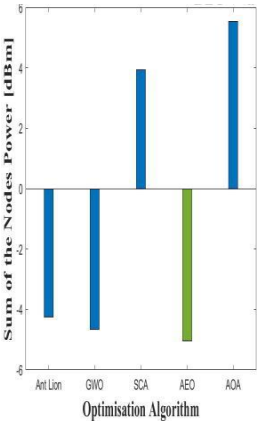
(d) Case no. 4



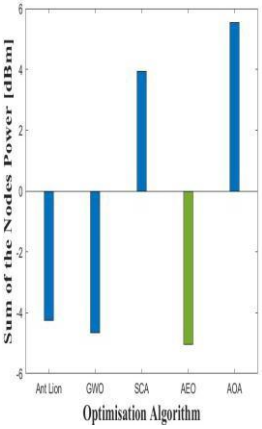
(e) [28]Case no. 5



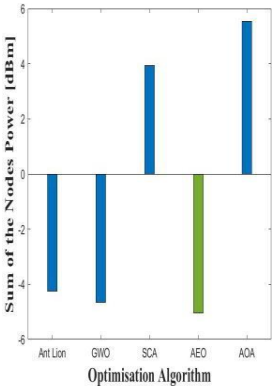
(f) Case no. 6



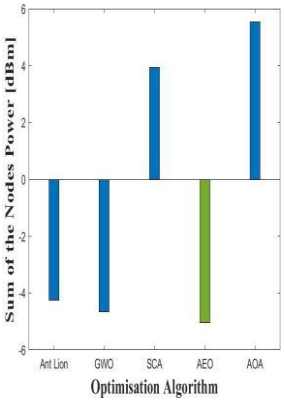
(g) Case no. 7



(h) Case no. 8



(i) Case no. 9



(j) Case no. 10

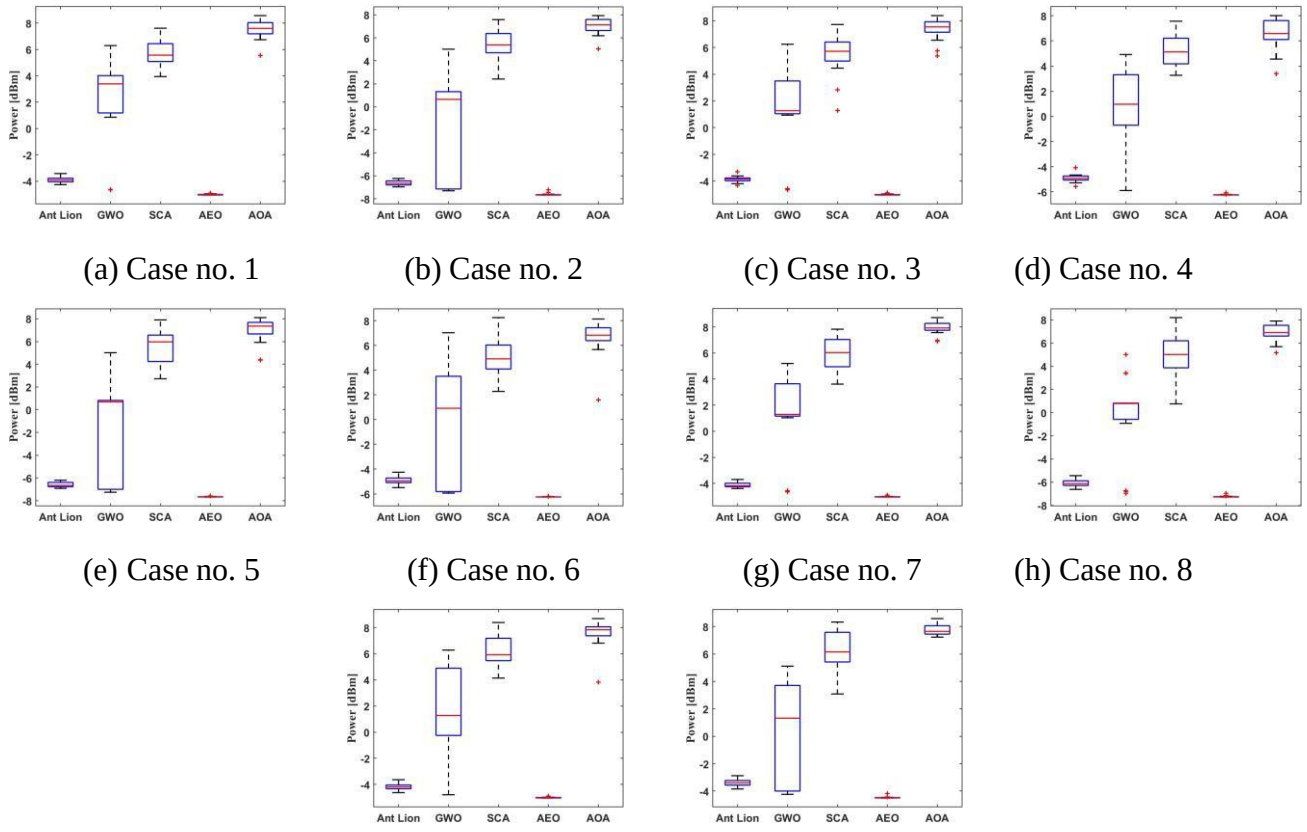
Fig 4.1: - Magnitude plot of optimizers.

The optimal value attained is presented in the form of a bar graph. The AEO optimizer, highlighted in green in Fig.4.1, provides the optimal solution in all ten cases. It was observed that the energy savings were greater in the case of AEO compared to the other algorithms. The AEO capability of optimization is superior to all others due to the fact that the production step guides the population fragments from one region to the best obtain region with greater speed as iterations increase. Some algorithms provided a positive. The negative value arises because the value of power here is less than 1 milliwatt. The results obtained are shown in Table II. The simplistic method (SM) is a method where all the nodes were assigned same transmission power [4]. This method provided a base cluster transmission power to judge optimizers performance. The average energy saved by AEO compared to the SM was around 6.4 dBm. While ALO was the second-best algorithm in terms of obtaining an optimal solution saving 5.66 dBm. AOA was the worst-performing algorithm out of all 5 optimizers.

4.3 Box plot analysis:

In descriptive statistics, a box plot [34] or boxplot is a method for graphically demonstrating the locality, spread and skewness groups of numerical data through their quartiles. In addition to the box on a box plot, there can be lines (which are called whiskers) extending from the box indicating variability outside the upper and lower quartiles, thus, the plot is also termed as the box-and-whisker plot and the box-and-whisker diagram. Outliers that differ significantly from the rest of the dataset may be plotted as individual points beyond the whiskers on the box-plot. Box plots are non-parametric: they display variation in samples of a statistical population without making any assumptions of the underlying statistical distribution (though Tukey's boxplot assumes symmetry for the whiskers and normality for their length). The spacings in each subsection of the box-plot indicate the degree of dispersion (spread) and skewness of the data, which are usually described using the five-number summary. In addition, the box-plot allows one to visually estimate various L-estimators, notably the interquartile range, midline, range, mid-range, and trimean. Although boxplots may seem primitive in comparison to a histogram or density plot, they have the advantage

of taking up less space, which is useful when comparing distributions between many groups or data sets. For all the test scenarios, the box plots for AEO are very narrow, the median lies lower as compared to other optimizer with no outliers. It is possible to conclude that AEO's value over 25 runs is very consistent. In Fig.4.2, AEO and ALO was observed as competitive with each other having little difference in optimal cluster transmission power. The box plot analysis depicts that when consistency is taken into account, ALO's medium is significantly higher than AEO's. ALO's Interquartile Range (IQR) is also high compared to AEO meaning more dispersion of optimal value. GWO IQR is large implying the value obtained over multiple runs is very inconsistent. Box plot analysis of shows, GWO, SCA and AOA are not even competitive to AEO and ALO. The box plot analysis of 10 different cases is given below: -



(a) Case no. 1 (b) Case no. 2 (c) Case no. 3 (d) Case no. 4

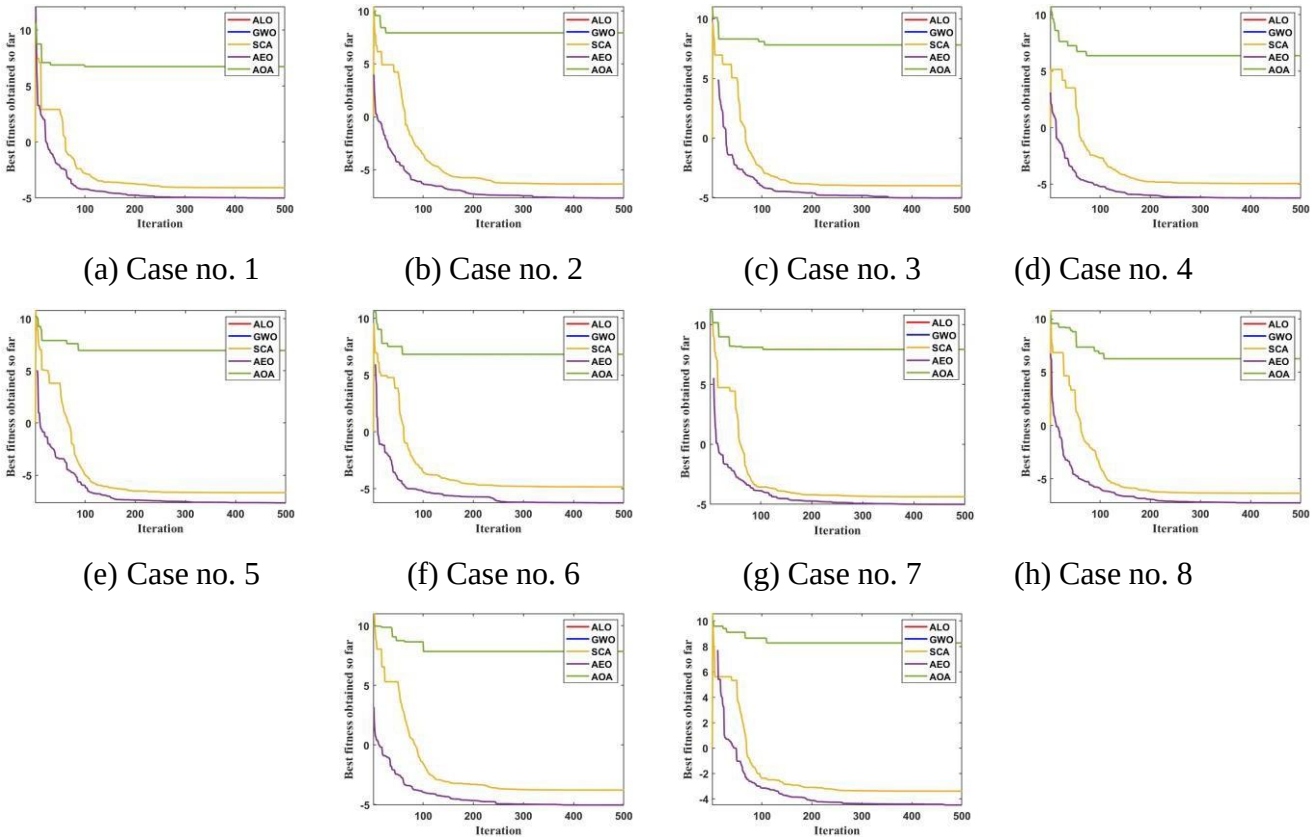
(e) Case no. 5 (f) Case no. 6 (g) Case no. 7 (h) Case no. 8

(i) Case no. 9 (j) Case no. 10

Fig 4.2: - Box plot of optimizers.

4.4 Convergence analysis:

The convergence curve analysis shows which optimization algorithm faster to reach the optimal solution graphically. Convergence is a very important criterion in WSNs. Even a small change in the optimal value can lead to huge differences in power consumption. The convergence curves are shown in Fig.6. The convergence curve was selected randomly out of 25 runs performed. AEO maintains a higher convergence value than other algorithms. It shows better exploration and exploitation capability than other optimizers in all 10 cases. On the contrary, despite providing a better solution, ALO has an inferior convergence rate, which leads to high transmission power. It clearly shows optimizer is lacking in exploitation capabilities. The convergence curves of ALO, GWO, and AOA reached the same solutions in all ten cases.



(a) Case no. 1 (b) Case no. 2 (c) Case no. 3 (d) Case no. 4

(e) Case no. 5 (f) Case no. 6 (g) Case no. 7 (h) Case no. 8

(i) Case no. 9 (j) Case no. 10

Fig 4.3: - Convergence plot of optimizers.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

We have successfully completed the simulation of the wireless sensor network in MATLAB. The base paper which we considered, used the simplistic and particle swarm optimization technique. The above method has been used to calculate different sensor nodes transmission power in order to save the sensor's energy while keeping all nodes connected. When compared to a constant power setup for the sensors, which is a common practice on the deployment of such networks, the PSO method could save, at least, 1 dBm of total network transmission power. However, it has not converged in all scenarios, thus requiring a further study on the fine-tuning of its parameters and its optimal number of runs.

We used five optimization algorithms other than PSO to calculate the minimum cluster transmission power from multiple nodes while keeping the network connected. The AEO algorithm gave the best result out of AOA, GWO, SCA, and ALO. Compared to simplistic method, average energy saved by AEO is 6.4 dBm. The superiority of the AEO optimizer for the mentioned model has been demonstrated using optimal value attained, box plots, and convergence curve analysis.

Modelling different types of antennas other than a simple isotropic antenna can be used to further investigate the brief model. To make it even more complicated, things like energy use, transmission rates, power use, security, and the differences between physical and medium access protocols could be taken into account.

References

[1] J. (. Kennedy, "Swarm intelligence.," *In Handbook of nature-inspired and innovative computing Springer*.

[2] S. (. Mirjalili, "Genetic algorithm.," *In Evolutionary algorithms and neural networks (pp. 43-55). Springer.*

[3] A. K. H. V. L. &. S. P. N. (. Qin, "Differential evolution algorithm with strategy adaptation for global numerical optimization.," *IEEE transactions on Evolutionary Computation,.*

[4] A. E. &. S. J. E. (. Eiben, "What is an evolutionary algorithm?," *In Introduction to evolutionary computing (pp. 25-48). Springer.*

[5] R. e. a. Miikkulainen, "Evolving deep neural networks," *Artificial intelligence in the age of neural networks and brain computing. Academic Press.*

[6] T. J. H. M. a. F. H. Elsken, "Neural architecture search: A survey," *The Journal of Machine Learning Research.*

[7] F. a. B. W. Marini, "Particle swarm optimization (PSO). A tutorial," *Chemometrics and Intelligent Laboratory Systems* .

[8] T. a. M. D. Stützle, "ACO algorithms for the traveling salesman problem.," *Evolutionary algorithms in engineering and computer science.*

[9] I. H. A. A. N. S. C. H. &. G. A. H. Ahmadianfar, "Ahmadianfar, I., Heidari, A. A., Noshadian, S., Chen, H., & Gandomi, A. H. (2022). INFO: An efficient optimization algorithm based on weighted mean of vectors.," *Ahmadianfar, I., Heidari, A. A., Noshadian, S., Chen, H., & Gandomi, A. H. (2022). INFO: An efficient optimization algorithm based on weighted mean of vectors. Expert Systems with Applications.*

[10] C. H. S. K. S. P. Prasad, " Cost-benefit analysis for optimal DG placement in distribution systems by using elephant herding," in *Renewables 6 (2),* (2019).

[11] A. M. C. R. D. M. &. R. H. S. Helmi, "Efficient and sustainable reconfiguration of distribution networks via metaheuristic optimization," *IEEE Transactions on Automation Science and Engineering, ,* 2021.

[12] M. E.-S. M. &. E.-M. M. AboGaleela, "A two level optimal DSM load shifting formulation using genetics algorithm case study: Residential loads," in *IEEE Power and Energy Society*, (2012, July). .

[13] I. K. H. Tan, " Power efficient data gathering and aggregation in wireless sensor networks," *ACM SIGMOD Record,* 2003.

[14] J. H. L. S. N. Li, "Design and analysis of an MST-based topology control algorithm," *IEEE Trans. Wirel. Commun,* 2005.

[15] I. K. I. S. H. Tan, "Computing localized power efficient data aggregation trees for sensor networks," *IEEE Trans. Paral. Dist. Syst. ,* 2011.

[16] G. T. J. Jaromczyk, "Relative neighborhood graphs and their relatives,," *IEEE,* 2002.

[17] Y. K. C. F. W. Liao, "Data aggregation in wireless sensor networks using ant colony algorithm," *Netw. Comp. Appl,* 2008.

[18] B. F. B. A. G. Mao, "Wireless sensor network localization techniques," *Comp. Netw.,* 2007.

[19] R. S. M. F. D. C. S. Madden, "Supporting aggregate queries over ad-hoc wireless sensor networks," in *IEEE Workshop on Mobile Computing Systems and Applications,* 2002.

[20] J. B. A. L. P. C. M. Sharaf, "Balancing energy efficiency and quality of aggregate data in sensor networks,," *VLDB J,* 2004.

[21] V. &. T. H. .. Rodoplu, " Minimum energy mobile wireless networks," *IEEE Journal on Selected Areas in Communications,* 1999.

[22] A. A. Ş. Y. A. F. P. &. I. M. Aziz, "A survey on distributed topology control techniques for extending the lifetime of battery powered wireless sensor networks," *IEEE Communications Surveysand Tutorials,* 2013.

[23] R. V. &. V. G. K. Kulkarni, " Particle swarm optimization in wireless-sensor networks: A brief survey," *IEEE Transactions on Systems, Man and Cybernetics Part C,* 2011.

[24] A. M. &. S. N. N. Gonsai, " Emerging Wireless Technologies for Fast Data," 2014.

[25] M. &. O. Abdelaal, "Recent energy-preservation endeavours for longlife wireless sensor networks: A concise survey," *Eleventh International Conference on Wireless and Optical Communications Network,* 2014.

[26] W. R. S. A. W. A. &. A. P. Heinzelman, "Energy-scalable algorithms and protocols for wireless microsensor networks," *IEEE International Conference on Acoustics,Speech, and Signal Processing.Proceedings,* 2000.

[27] L. D. P. R. J. J. P. C. &. C. M. Mendes, "A cross-layer sleep and rate adaptation mechanism for slotted ALOHA wireless sensor networks," *International Conference on Information and Communication Technology Convergence, ICTC,* 2010.

[28] D. H. a. W. G. M. Wolpert, "No free lunch theorems for optimization.," *IEEE transactions on evolutionary computation,* pp. 67-82, 1997.

[29] S. Mirjalili, "SCA: a sine cosine algorithm for solving optimization problems," *Knowledge-based systems 96,* pp. 120-133, 2016.

[30] S. Mirjalili, "The ant lion optimizer," *Advances in engineering software 83,* pp. 80-98, 2015.

[31] W. L. W. a. Z. Z. Zhao, "Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm," *eural Computing and Applications 32* , pp. 9383-9425, 2020.

[32] H. O. B.-H. a. X. C. Rezaei, "Grey wolf optimization (GWO) algorithm," *Advanced optimization by nature-inspired algorithms,* pp. 81-91, 2018.

[33] L. e. a. Abualigah, “The arithmetic optimization algorithm.,” *Computer methods in applied mechanics and engineering 376,* p. 113609, 2021.

[34] D. F. R. A. P. a. J. S. K. Williamson, “ The box plot: a simple visual method to interpret data.,” *Annals of internal medicine ,* pp. 916-921, 1989.

PUBLICATION

- Anshul Kumar Yadav, Akshita Sharma, Anurag Yadav, Ankit Vijayvargiya, Akash Saxena, & Rajesh Kumar, “Optimization Scheme for Power Transmission in Wireless Sensor Network”, International Conference on Power, Instrumentation, Energy, and Control (PIECON-2023), Feb 10-12, 2023.

Appendix

1) Code: -

- **Main**

```
close all
clear all
clc

SearchAgents_no=30; % Number of search agents

Function_name='F24'; % Name of the test function that can be from F1 to F23
(Table 1,2,3 in the paper)
                % F24 for WSN test case.
Max_iteration=500; % Maximum numbef of iterations

% Load details of the selected benchmark function
[lb,ub,dim,fobj]=Get_Functions_details(Function_name);

% for test case
for t=1:25
for cen=1:10
 cenario = strcat('space',int2str(cen));
 load(cenario)
 nomeArquivo = strcat(cenario,'.t2');

fid = fopen(nomeArquivo,'wt');
fprintf(fid,'##################Start of the
training##################\n\n');
fprintf(fid,'Cenario = %s\n\n',cenario);

[Gwo_score(t,cen),Gwo_pos(cen,:),GWO_cg_curve(t,cen,:)]=GWO(SearchAg
ents_no,Max_iteration,lb,ub,dim,fobj,X,fid);
[AEO_score(t,cen),AEO_pos(cen,:),AEO_cg_curve(t,cen,:)]=AEO(SearchAge
nts_no,Max_iteration,lb,ub,dim,fobj,X,fid);
[FDA_score(t,cen),FDA_pos(cen,:),FDA_cg_curve(t,cen,:)]=ALO(SearchAge
nts_no,Max_iteration,lb,ub,dim,fobj,X,fid);
[SCA_score(t,cen),SCA_pos(cen,:),SCA_cg_curve(t,cen,:)]=SCA(SearchAge
nts_no,Max_iteration,lb,ub,dim,fobj,X,fid);
```

```
[AOA_score(t,cen),AOA_pos(cen,:),AOA_cg_curve(t,cen,:)]=AOA(SearchAg
ents_no,Max_iteration,lb,ub,dim,fobj,X,fid);

end
end
% save m-file
cenario = strcat('run');
save(cenario);
```

- **Connect report**

```
function [soma] = connect_report(X,power_vector)



N = size(X,2);


freq = 915;



cost = zeros(N);

for l=1:N
   for c=1:N

      if l == c
         cost(l,c) = -Inf;
      else
         cost(l,c) = FRISS(1,1,freq, X(:,l), X(:,c));
      end

   end
end



for i = 1:N
   power_vector_watt (i) = 10.^((power_vector(i)./10)-3);
```

```
end

PR = zeros(N);

for l = 1: N
   for c = 1 : N
      PR(l,c) = power_vector(l) + cost(l,c);
   end
end

A = zeros(N);
for l = 1:N
   for c = 1: N

      if PR(l,c) >= -60
         A (l,c) = 1;
      else
         A (l,c) = 0;
      end
   end
end

for i=1:N
   D(i) = sum(A(i,:));
end

connect = 1;
for i = 1:N
   connect = D(i).*connect;
end


if connect == 0
   soma = Inf;

else


   for i = 1:N
```

```
        for j = 1:N

            if i == j
                L(i,j) = D(i);
            else
                if  A(i,j) == 1
                    L(i,j) = -1;
                else

                L(i,j) = 0;
                end
            end
        end

        autovetor = eig(L);

        if autovetor(2) > 0
            soma = 10.*log10((sum(power_vector_watt))/(1e-3));
        else
            soma = Inf;
        end
    end

    end
```

- **FRISS**

```
    function [ATTdb] = friss(Gt, Gr, freq, Txp, Rxp)

    c = (3.*10.^8);
    lambda = c ./ (freq.*10.^6);

    dist = (((Rxp(1) - Txp(1)).^2) + ((Rxp(2) - Txp(2)).^2)).^0.5;

    ATT = Gt.*Gr.*(lambda./(4.*pi.*dist)).^2;
    ATTdb = 10.*log10(ATT);
    End
```

- **Grey Wolf Optimizer**

```
    % Grey Wolf Optimizer
```

```
function
[Alpha_score,Alpha_pos,Convergence_curve]=GWO(SearchAgents_n
o,Max_iter,lb,ub,dim,fobj,X,fid)

% initialize alpha, beta, and delta_pos
Alpha_pos=zeros(1,dim);
Alpha_score=inf; %change this to -inf for maximization problems

Beta_pos=zeros(1,dim);
Beta_score=inf; %change this to -inf for maximization problems

Delta_pos=zeros(1,dim);
Delta_score=inf; %change this to -inf for maximization problems

%Initialize the positions of search agents
Positions=initialization(SearchAgents_no,dim,ub,lb);

Convergence_curve=zeros(1,Max_iter);

l=0;% Loop counter

% Main loop
while l<Max_iter
   for i=1:size(Positions,1)

      % Return back the search agents that go beyond the boundaries of
the search space
      Flag4ub=Positions(i,:)>ub;
      Flag4lb=Positions(i,:)<lb;

Positions(i,:)=(Positions(i,:).*(~(Flag4ub+Flag4lb)))+ub.*Flag4ub+lb.
*Flag4lb;

      % Calculate objective function for each search agent
      fitness=fobj(X,Positions(i,:));

      % Update Alpha, Beta, and Delta
      if fitness<Alpha_score
         Alpha_score=fitness; % Update alpha
```

```
            Alpha_pos=Positions(i,:);
        end

        if fitness>Alpha_score && fitness<Beta_score
            Beta_score=fitness; % Update beta
            Beta_pos=Positions(i,:);
        end

        if fitness>Alpha_score && fitness>Beta_score &&
fitness<Delta_score
            Delta_score=fitness; % Update delta
            Delta_pos=Positions(i,:);
        end
    end


    a=2-l*((2)/Max_iter); % a decreases linearly fron 2 to 0

    % Update the Position of search agents including omegas
    for i=1:size(Positions,1)
        for j=1:size(Positions,2)

            r1=rand(); % r1 is a random number in [0,1]
            r2=rand(); % r2 is a random number in [0,1]

            A1=2*a*r1-a; % Equation (3.3)
            C1=2*r2; % Equation (3.4)

            D_alpha=abs(C1*Alpha_pos(j)-Positions(i,j)); % Equation
(3.5)-part 1
            X1=Alpha_pos(j)-A1*D_alpha; % Equation (3.6)-part 1

            r1=rand();
            r2=rand();

            A2=2*a*r1-a; % Equation (3.3)
            C2=2*r2; % Equation (3.4)
```

```
            D_beta=abs(C2*Beta_pos(j)-Positions(i,j)); % Equation (3.5)-
part 2
            X2=Beta_pos(j)-A2*D_beta; % Equation (3.6)-part 2

            r1=rand();
            r2=rand();

            A3=2*a*r1-a; % Equation (3.3)
            C3=2*r2; % Equation (3.4)

            D_delta=abs(C3*Delta_pos(j)-Positions(i,j)); % Equation (3.5)-
part 3
            X3=Delta_pos(j)-A3*D_delta; % Equation (3.5)-part 3

            Positions(i,j)=(X1+X2+X3)/3;% Equation (3.7)

        end
    end
    l=l+1;
    Convergence_curve(l)=Alpha_score;
    if(l==1 || rem(l,50)==0)
        sprintf('Iter - %4d | Alpha_score = %f\n',l,Alpha_score)
        fprintf(fid,'Iter - %4d | Alpha_score = %f\n',l,Alpha_score);
    end
end
```

- **Ant Lion Optimizer**

```
function
[Elite_antlion_fitness,Elite_antlion_position,Convergence_curve]=AL
O(N,Max_iter,lb,ub,dim,fobj,y,fid)

% Initialize the positions of antlions and ants
antlion_position=initialization(N,dim,ub,lb);
ant_position=initialization(N,dim,ub,lb);

% Initialize variables to save the position of elite, sorted antlions,
% convergence curve, antlions fitness, and ants fitness
Sorted_antlions=zeros(N,dim);
Elite_antlion_position=zeros(1,dim);
Elite_antlion_fitness=inf;
```

```
Convergence_curve=zeros(1,Max_iter);
antlions_fitness=zeros(1,N);
ants_fitness=zeros(1,N);

% Calculate the fitness of initial antlions and sort them
for i=1:size(antlion_position,1)
   antlions_fitness(1,i)=fobj(y,antlion_position(i,:));
end

[sorted_antlion_fitness,sorted_indexes]=sort(antlions_fitness);

for newindex=1:N

Sorted_antlions(newindex,:)=antlion_position(sorted_indexes(newinde
x),:);
end

Elite_antlion_position=Sorted_antlions(1,:);
Elite_antlion_fitness=sorted_antlion_fitness(1);

% Main loop start from the second iteration since the first iteration
% was dedicated to calculating the fitness of antlions
Current_iter=2;
while Current_iter<Max_iter+1

   % This for loop simulate random walks
   for i=1:size(ant_position,1)
      % Select ant lions based on their fitness (the better anlion the
higher chance of catching ant)

Rolette_index=RouletteWheelSelection(1./sorted_antlion_fitness);
      if Rolette_index==-1
         Rolette_index=1;
      end

      % RA is the random walk around the selected antlion by rolette
wheel
      RA=Random_walk_around_antlion(dim,Max_iter,lb,ub,
Sorted_antlions(Rolette_index,:),Current_iter);
```

```
        % RA is the random walk around the elite (best antlion so far)
        [RE]=Random_walk_around_antlion(dim,Max_iter,lb,ub,
Elite_antlion_position(1,:),Current_iter);

        ant_position(i,:)= (RA(Current_iter,:)+RE(Current_iter,:))/2; %
Equation (2.13) in the paper
    end

    for i=1:size(ant_position,1)

        % Boundar checking (bring back the antlions of ants inside search
        % space if they go beyoud the boundaries
        Flag4ub=ant_position(i,:)>ub;
        Flag4lb=ant_position(i,:)<lb;

ant_position(i,:)=(ant_position(i,:).*(~(Flag4ub+Flag4lb)))+ub.*Flag4
ub+lb.*Flag4lb;

        ants_fitness(1,i)=fobj(y,ant_position(i,:));

    end

    % Update antlion positions and fitnesses based of the ants (if an ant
    % becomes fitter than an antlion we assume it was cought by the
antlion
    % and the antlion update goes to its position to build the trap)
    double_population=[Sorted_antlions;ant_position];
    double_fitness=[sorted_antlion_fitness ants_fitness];

    [double_fitness_sorted I]=sort(double_fitness);
    double_sorted_population=double_population(I,:);

    antlions_fitness=double_fitness_sorted(1:N);
    Sorted_antlions=double_sorted_population(1:N,:);

    % Update the position of elite if any antlinons becomes fitter than it
    if antlions_fitness(1)<Elite_antlion_fitness
        Elite_antlion_position=Sorted_antlions(1,:);
```

```
        Elite_antlion_fitness=antlions_fitness(1);
    end

    % Keep the elite in the population
    Sorted_antlions(1,:)=Elite_antlion_position;
    antlions_fitness(1)=Elite_antlion_fitness;

    % Update the convergence curve
    Convergence_curve(Current_iter)=Elite_antlion_fitness;

   if(Current_iter==1 || rem(Current_iter,50)==0)
        sprintf('Iter - %4d | ALo =
%f\n',Current_iter,Elite_antlion_fitness)
        fprintf(fid,'Iter - %4d | ALO =
%f\n',Current_iter,Elite_antlion_fitness);
    end
    Current_iter=Current_iter+1;
end
```

- **Artificial Ecosystem Optimzer**

```
function
[BestF,BestX,HisBestFit]=AEO(nPop,MaxIt,Low,Up,Dim,fobj,X,fid)


% FunIndex: Index of function.
% MaxIt: The maximum number of iterations.
% PopSize: The size of population.
% PopPos: The position of population.
% PopFit: The fitness of population.
% Dim: The dimensionality of prloblem.
% C: The consumption factor.
% D: The decomposition factor.
% BestX: The best solution found so far.
% BestF: The best fitness corresponding to BestX.
% HisBestFit: History best fitness over iterations.
% Low: The low bound of search space.
% Up: The up bound of search space.
```

```
PopPos=zeros(nPop,Dim);
PopFit=zeros(1,nPop);
for i=1:nPop
   PopPos(i,:)=rand(1,Dim).*(Up-Low)+Low;
   PopFit(i)=fobj(X,PopPos(i,:));
end


[~, indF]=sort(PopFit,'descend');
PopPos=PopPos(indF,:);
PopFit=PopFit(indF);
BestF=PopFit(end);
BestX=PopPos(end,:);

HisBestFit=zeros(MaxIt,1);
Matr=[1,Dim];

for It=1:MaxIt
   r1=rand;
   a=(1-It/MaxIt)*r1;
   xrand=rand(1,Dim).*(Up-Low)+Low;
   newPopPos(1,:)=(1-a)*PopPos(nPop,:)+a*xrand; %equation (1)

   u=randn(1,Dim);
   v=randn(1,Dim);
   C=1/2*u./abs(v); %equation (4)
   newPopPos(2,:)=PopPos(2,:)+C.*(PopPos(2,:)-newPopPos(1,:));
%equation (6)

   for i=3:nPop
      u=randn(1,Dim);
      v=randn(1,Dim);
      C=1/2*u./abs(v);
      r=rand;
      if r<1/3
         newPopPos(i,:)=PopPos(i,:)+C.*(PopPos(i,:)-newPopPos(1,:));
%equation (6)
      else
         if  1/3<r && r<2/3
```

```
            newPopPos(i,:)=PopPos(i,:)+C.*(PopPos(i,:)-
PopPos(randi([2 i-1]),:)); %equation (7)
        else
            r2=rand;
            newPopPos(i,:)=PopPos(i,:)+C.*(r2.*(PopPos(i,:)-
newPopPos(1,:))+(1-r2).*(PopPos(i,:)-PopPos(randi([2  i-1]),:)));
%equation (8)
        end
    end
end

for i=1:nPop
    newPopPos(i,:)=SpaceBound(newPopPos(i,:),Up,Low);
    newPopFit(i)= fobj(X,newPopPos(i,:));
    if newPopFit(i)<PopFit(i)
        PopFit(i)=newPopFit(i);
        PopPos(i,:)=newPopPos(i,:);
    end
end

[~, indOne]=min(PopFit);
for i=1:nPop
    r3=rand;   Ind=round(rand)+1;

newPopPos(i,:)=PopPos(indOne,:)+3*randn(1,Matr(Ind)).*((r3*randi([
1 2])-1)*PopPos(indOne,:)-(2*r3-1)*PopPos(i,:)); %equation (9)
end

for i=1:nPop
    newPopPos(i,:)=SpaceBound(newPopPos(i,:),Up,Low);
    newPopFit(i)=fobj(X,newPopPos(i,:));
    if newPopFit(i)<PopFit(i)
        PopPos(i,:)=newPopPos(i,:);
        PopFit(i)=newPopFit(i);
    end
end

[~,indF]=sort(PopFit,'descend');
PopPos=PopPos(indF,:);
```

```
            PopFit=PopFit(indF);

            if PopFit(end)<BestF
               BestF=PopFit(end);
               BestX=PopPos(end,:);
            end

            HisBestFit(It)=BestF;
            if(It==1 || rem(It,50)==0)
               sprintf('Iter - %4d | _score = %f\n',It,BestF)
               fprintf(fid,'Iter - %4d | Alpha_score = %f\n',It,BestF);
            end
         end
```

- **Sine Cosine Optimizer**

```
function [Destination_fitness,Destination_position,Convergence_curve]=SCA(N,Max_iteration,lb,ub,dim,fobj,y,fid)

display('SCA is optimizing your problem');

%Initialize the set of random solutions
X=initialization(N,dim,ub,lb);

Destination_position=zeros(1,dim);
Destination_fitness=inf;

Convergence_curve=zeros(1,Max_iteration);
Objective_values = zeros(1,size(X,1));

% Calculate the fitness of the first set and find the best one
for i=1:size(X,1)
   Objective_values(1,i)=fobj(y,X(i,:));
   if i==1
      Destination_position=X(i,:);
      Destination_fitness=Objective_values(1,i);
   elseif Objective_values(1,i)<Destination_fitness
      Destination_position=X(i,:);
      Destination_fitness=Objective_values(1,i);
   end
```

```
    All_objective_values(1,i)=Objective_values(1,i);
end

%Main loop
t=2; % start from the second iteration since the first iteration was dedicated to
calculating the fitness
while t<=Max_iteration

    % Eq. (3.4)
    a = 1;
    Max_iteration = Max_iteration;
    r1=a-t*((a)/Max_iteration); % r1 decreases linearly from a to 0

    % Update the position of solutions with respect to destination
    for i=1:size(X,1) % in i-th solution
        for j=1:size(X,2) % in j-th dimension

            % Update r2, r3, and r4 for Eq. (3.3)
            r2=(2*pi)*rand();
            r3=2*rand;
            r4=rand();

            % Eq. (3.3)
            if r4<0.5
                % Eq. (3.1)
                X(i,j)= X(i,j)+(r1*sin(r2)*abs(r3*Destination_position(j)-X(i,j)));
            else
                % Eq. (3.2)
                X(i,j)= X(i,j)+(r1*cos(r2)*abs(r3*Destination_position(j)-X(i,j)));
            end

        end
    end

    for i=1:size(X,1)

        % Check if solutions go outside the search spaceand bring them back
        Flag4ub=X(i,:)>ub;
```

```
        Flag4lb=X(i,:)<lb;
        X(i,:)=(X(i,:).*(~(Flag4ub+Flag4lb)))+ub.*Flag4ub+lb.*Flag4lb;

        % Calculate the objective values
        Objective_values(1,i)=fobj(y,X(i,:));

        % Update the destination if there is a better solution
        if Objective_values(1,i)<Destination_fitness
            Destination_position=X(i,:);
            Destination_fitness=Objective_values(1,i);
        end
    end

    Convergence_curve(t)=Destination_fitness;

    % Display the iteration and best optimum obtained so far
    if mod(t,50)==0
        display(['At iteration ', num2str(t), ' the optimum is ',
num2str(Destination_fitness)]);
    end

    % Increase the iteration counter
    t=t+1;
    if(t==1 || rem(t,50)==0)
        sprintf('Iter - %4d | Destination_fitness = %f\n',t,Destination_fitness)
        fprintf(fid,'Iter - %4d | Destination_fitness = %f\n',t,Destination_fitness);
    end
end
```

- **Arithmetic Optimizer**

```
function
[Best_FF,Best_P,Conv_curve]=AOA(N,M_Iter,LB,UB,Dim,F_obj,y,fid)
display('AOA Working');
%Two variables to keep the positions and the fitness value of the best-
obtained solution

Best_P=zeros(1,Dim);
Best_FF=inf;
Conv_curve=zeros(1,M_Iter);
```

```
%Initialize the positions of solution
X=initialization(N,Dim,UB,LB);
Xnew=X;
Ffun=zeros(1,size(X,1));% (fitness values)
Ffun_new=zeros(1,size(Xnew,1));% (fitness values)

MOP_Max=1;
MOP_Min=0.2;
C_Iter=1;
Alpha=5;
Mu=0.499;


for i=1:size(X,1)
   Ffun(1,i)=F_obj(y,X(i,:));  %Calculate the fitness values of solutions
   if Ffun(1,i)<Best_FF
      Best_FF=Ffun(1,i);
      Best_P=X(i,:);
   end
end



while C_Iter<M_Iter+1  %Main loop
   MOP=1-((C_Iter)^(1/Alpha)/(M_Iter)^(1/Alpha));  % Probability Ratio
   MOA=MOP_Min+C_Iter*((MOP_Max-MOP_Min)/M_Iter); %Accelerated
function

   %Update the Position of solutions
   for i=1:size(X,1)  % if each of the UB and LB has a just value
      for j=1:size(X,2)
        r1=rand();
         if (size(LB,2)==1)
            if r1<MOA
               r2=rand();
               if r2>0.5
                  Xnew(i,j)=Best_P(1,j)/(MOP+eps)*((UB-LB)*Mu+LB);
               else
                  Xnew(i,j)=Best_P(1,j)*MOP*((UB-LB)*Mu+LB);
```

```
            end
        else
            r3=rand();
            if r3>0.5
                Xnew(i,j)=Best_P(1,j)-MOP*((UB-LB)*Mu+LB);
            else
                Xnew(i,j)=Best_P(1,j)+MOP*((UB-LB)*Mu+LB);
            end
        end
    end


    if (size(LB,2)~=1)   % if each of the UB and LB has more than one
value
        r1=rand();
        if r1<MOA
            r2=rand();
            if r2>0.5
                Xnew(i,j)=Best_P(1,j)/(MOP+eps)*((UB(j)-
LB(j))*Mu+LB(j));
            else
                Xnew(i,j)=Best_P(1,j)*MOP*((UB(j)-LB(j))*Mu+LB(j));
            end
        else
            r3=rand();
            if r3>0.5
                Xnew(i,j)=Best_P(1,j)-MOP*((UB(j)-LB(j))*Mu+LB(j));
            else
                Xnew(i,j)=Best_P(1,j)+MOP*((UB(j)-LB(j))*Mu+LB(j));
            end
        end
    end

end

Flag_UB=Xnew(i,:)>UB; % check if they exceed (up) the boundaries
Flag_LB=Xnew(i,:)<LB; % check if they exceed (down) the boundaries
```

```
Xnew(i,:)=(Xnew(i,:).*(~(Flag_UB+Flag_LB)))+UB.*Flag_UB+LB.*Flag_L
B;

    Ffun_new(1,i)=F_obj(y,Xnew(i,:));  % calculate Fitness function
    if Ffun_new(1,i)<Ffun(1,i)
       X(i,:)=Xnew(i,:);
       Ffun(1,i)=Ffun_new(1,i);
    end
    if Ffun(1,i)<Best_FF
    Best_FF=Ffun(1,i);
    Best_P=X(i,:);
    end

  end


  %Update the convergence curve
  Conv_curve(C_Iter)=Best_FF;
  C_Iter=C_Iter+1;  % incremental iteration

end
```