OPTIMIZING THE MOVEMENT PATH IN A NETWORK OF GROUND AND AERIAL MOBILE ROBOTS IN THE FIELD OF COMMUNICATION AND TRANSPORTATION

Project Report

Field of Electronic and Computer Engineering

fulfillment of the requirements for the degree of MEng

Student's Name

Reza Doostniae

Supervisor

Dr. Amirali Baniasadi

August 2023

Abstract

Numerous navigation and routing systems exist in the field of mobile robotics. This project aims to leverage the strengths of existing systems and incorporate the bat optimization algorithm to enhance the performance of a network comprising ground and aerial mobile robots. A novel algorithm has been developed to automatically detect and optimize the movement path within this network, focusing on improving communication and transportation capabilities.

The selection of the bat optimization algorithm was based on specific reasons, including the similarity between the movement mechanism and data transmission of aerial robots with the flight pattern of bats, as well as its algorithmic features. This choice was made among various meta-heuristic algorithms such as genetics and ant colony optimization.

Keywords: routing, network, meta-heuristic algorithms, aerial and ground robots.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

DEDICATION

I would like to express my heartfelt gratitude to my supervisor, Dr. Amirali Baniasdi, for his invaluable guidance and unwavering support throughout this research. His expertise and mentorship have greatly contributed to the success of this thesis.

I am also deeply grateful to my wife for her constant support, understanding, and encouragement. Her presence and belief in me have been a tremendous source of inspiration.

I am truly fortunate to have such an exceptional supervisor and a loving wife who have played instrumental roles in my academic and personal growth. Their guidance and unwavering support have been invaluable, and I am profoundly grateful for their presence in my life.

I extend my sincere appreciation for their belief in me and their contributions to my academic achievements. Their love, support, and encouragement have made a significant impact, and I am forever grateful for their unwavering belief in my abilities.

1 Introduction

Motion sensors and routing systems play a crucial role in digital systems and industries related to the Internet of Things, especially in mechatronics, communication, and land and air transportation [1]. These systems, along with technologies such as image processing and artificial intelligence, have enabled the development of numerous practical applications [2]

In the field of communication and transportation, autonomous or remotely controlled mobile robots are extensively used [3]. To ensure their successful operation and accomplishment of assigned tasks, reliable orientation and routing systems are essential. Robots need to be capable of creating a program that detects an optimal and smooth movement path, minimizing both distance and fluctuations in speed. The problem of planning robot movement has been a subject of research in electrical, computer, and robotics sciences since the late 1970s, reflecting its importance in various engineering disciplines [4].

The applications of finding optimal or safe movement paths extend beyond these sciences and have implications in many engineering fields. Examples include planning the movement of lift trucks within factory premises [5], determining the shortest routes for private and public vehicles in urban areas [6], and designing optimal paths for connecting electronic components on printed circuit boards to minimize occupied space.

Existing methods for guiding robots can be categorized into three main types: hierarchical, behavioral, and hybrid. Similarly, the methods for planning robot movement can be classified based on the availability of information. The first type relies on non-real-time information, utilizing pre-existing maps and prior knowledge. The second type involves real-time planning,

where the robot acquires information about its surroundings through sensors during the planning process [7].

To find an optimal path from the starting point to the target point while avoiding obstacles and efficiently moving data packets, employing a meta-heuristic algorithm inspired by nature can be a suitable and effective solution. Such an algorithm can provide a high level of confidence in addressing this challenge [8].

The significance of this issue becomes evident when network robots, employed to streamline operations in transportation and communication industries, fail to follow the correct path due to path recognition weaknesses and lack of obstacle identification. Inaccurate execution of tasks not only results in lower accuracy but also increases time consumption and the risk of potential damage.

This project investigates various types of motion sensors used in mobile robots, as well as routing and orientation algorithms. Specifically, the behavior and application of kinematic sensors, multi-axis accelerometers, and gyroscopes within the robot network are examined. Communication and movement algorithms, along with their optimization, are discussed in relation to these sensors.

The main focus of this research is to develop a comprehensive algorithm for determining the optimal movement path in a network of mobile robots. In this movement path, when a network consists of both ground and aerial robots, one robot is designated as the service and support station, serving as a reference or mother robot. Other robots follow this reference robot closely, constantly transmitting motion data such as coordinate position and angular velocity. Based on the motion sensor data, each robot announces its status and utilizes the optimization algorithm to determine

the optimal movement path. This routing algorithm enables network robots to complete their assigned tasks in less time, fostering better cooperation and achieving improved outcomes.

In summary, this project aims to provide a comprehensive algorithm for determining the optimal movement path within a network of mobile robots. By employing motion sensors, communication, and optimization algorithms, the network can efficiently navigate and accomplish tasks, resulting in reduced time consumption and enhanced performance.

1.1 Problem Statement

In the field of communication and transportation, the use of mobile robots, both on the ground and in the air, has gained significant attention. These robots are employed to perform various tasks autonomously or under remote control. However, to ensure their successful operation and efficient task execution, a reliable orientation and routing system is crucial.

The problem arises when the existing navigation and routing systems fail to provide optimal and smooth movement paths for the network of ground and aerial mobile robots. Inaccurate path detection, obstacles recognition weaknesses, and lack of efficient coordination among the robots lead to suboptimal performance, increased time consumption, and the possibility of potential damage.

Therefore, there is a need to develop an algorithm that optimizes the movement path in a network of ground and aerial mobile robots, considering factors such as path length, smoothness of movement, and efficient utilization of available resources. This algorithm should leverage the strengths of existing systems while incorporating meta-heuristic algorithms inspired by nature, such as the bat optimization algorithm, to enhance the routing and coordination capabilities of the network robots.

The objective of this thesis is to address this problem by proposing a comprehensive algorithm that enables the network robots to detect the optimal movement path in real-time, considering their individual capabilities, motion sensor data, and coordination with the reference or mother robot. The algorithm should optimize the routing process, minimize time consumption, and improve the overall performance of the network of mobile robots in communication and transportation applications.

By developing an efficient algorithm, this research aims to contribute to the advancement of navigation and routing systems in the field of electronic and computer engineering, enhancing the capabilities of mobile robots and promoting their effective utilization in various industries and domains.

1.2 Research approach

The table provides an overview of the research methodology used in this study, summarizing the key components of the research approach.

Table 1.1: Research approach

Research Methodology Components	Description
Research Approach	Quantitative research approach
Sampling Technique	Non-probabilistic purposive sampling
Data Collection	Benchmark functions and performance metrics obtained from simulation experiments
Experimental Procedure	Implementation of Bat Algorithm and Genetic Algorithm using a programming language
Data Analysis	Statistical analysis using performance comparison, mean analysis, and graphical representation
Limitations	Constraints in data availability and diversity for training algorithms

2 Literature Review

In this project, we have conducted investigations on various types of motion sensors in mobile robots, as well as routing and orientation algorithms. Due to the wide range of sensors in this category, we have examined the behavior of kinematic sensors, multi-axis accelerometers, and gyroscope sensors. We have also explored their applications in the robot network. Additionally, we have discussed the relevant communication and movement algorithms, along with their optimization techniques.

The primary objective is to find the optimal movement path while ensuring obstacle avoidance. This means selecting a path that avoids collisions with fixed objects as well as moving objects. To achieve safe and reliable navigation, obstacle avoidance algorithms are applied after extracting obstacle shapes and determining the need for avoidance.

Another aspect addressed in this project is the problem of node placement in wireless sensor networks. This involves determining the geographic coordinates of each device with an unknown location in the deployment area. In this chapter, we present existing algorithms for routing, orientation, and finding the optimal path. Furthermore, we examine the limitations of the current systems.

It is worth mentioning that an optimization algorithm is described to evaluate the accuracy of the node localization problem in wireless sensor networks. This algorithm aims to improve the precision and efficiency of the localization process.

2.1 Review of Routing Systems

In the field of locating, orientation, and distance calculation, there are various systems and methods that operate based on the time it takes for a wave to travel and return. However, despite the facilities provided, there are circumstances where their usage becomes impractical.

Routing technology plays a crucial role in assisting vehicle drivers in multiple ways. Its primary application is to find the best path of movement in terms of path length, smoothness, and obstacle traversal. Ensuring accurate navigation and reaching the desired endpoint are paramount in all these systems [9]. Additionally, recent advancements have introduced additional features such as monitoring the internal systems of the vehicle, detecting road accidents to alert the driver, and providing emergency notifications in case of an accident. Routing technology is widely employed not only in private cars and public vehicles but also in certain commercial vehicles like trains and ships [10]. However, the most common use of route finders in private cars is to identify the optimal route between two points [11].

On the other hand, tracking devices serve a distinct purpose compared to routing systems. Their primary goal is to collect data pertaining to the geographic location, direction, and speed of the transmitter. This data is then analyzed to provide real-time additional information to the user. Unlike routing systems, tracking devices can record and store route and geographic location information for future investigations. Tracking devices have a broader range of applications compared to routing systems. For instance, they are extensively used by public transport companies, goods transit companies, government organizations such as the police, taxi and car rental companies, private individuals for vehicle safety, families for protecting children, elderly individuals, and even pets, among others [12].

Overall, tracking devices have wider applications and functionalities compared to routing systems, making them essential tools in various industries.

2.1.1 GPS System

In router systems, the Global Positioning System (GPS) is considered a prominent positioning system Initially developed for military tracking purposes, the US military deployed the first tracking satellite into Earth's orbit in 1978. In subsequent years, the United States popularized this technology, making it accessible to the public. Currently, there are approximately 24 satellites, known as Navastar, orbiting the Earth at fixed distances. These satellites are organized into six orbits. It is worth noting that these satellites have a lifespan of approximately 10 years and are periodically replaced with new ones. Each satellite completes two orbits around the Earth daily [13].

GPS devices can function in all weather conditions and at any time. When a GPS device is turned on, it detects and calculates the user's position based on the signals received from three satellites closest to their location. The calculated position is then displayed on the device's map. If the device can establish communication with four satellites, the accuracy of the position determination improves, reducing the margin of error. GPS devices offer various features, including calculating movement speed, trip duration, distance to the destination, geographic coordinates (longitude and latitude), sunrise and sunset times, local time, and the number of satellites within range. As a result, GPS devices find utility in a wide range of activities such as fishing, gliding, skiing, ground and military operations, sailing, driving, rescue missions, group trips and nature excursions, car racing, and more.

While GPS remains the most common wireless positioning technology, it relies on satellite communication, which can be challenging in closed environments. Moreover, its effectiveness may be limited in personal and independent projects, and its inherent errors cannot be overlooked.

2.1.2 Lidar System

The Lidar system [14] utilizes laser technology to determine distances by calculating the angle of the reflected laser beam. It offers high accuracy and speed; however, it has certain limitations. Lidar systems are unable to accurately calculate distances for transparent objects and shiny surfaces like glass and metals. Additionally, the use of advanced radar technology is only effective for very long distances. Furthermore, these systems encounter difficulties when there are obstacles obstructing the line of sight between the sensor and the target.

2.2 Distance measurement based on the laws of the Doppler effect

The technology known as Distance Finding based on the Doppler Effect (DFDE) [15] utilizes the principles of the Doppler effect to determine distances. The Doppler effect, observed in sound and light waves, describes the change in frequency when a receiver moves towards or away from a stationary source. When the receiver approaches the source, it experiences an increase in frequency, and conversely, when it moves away, the frequency decreases. By analyzing these frequency changes, the distance between the transmitter and receiver can be calculated. It is important to note that the Doppler effect can only be applied when either the transmitter or the receiver is in motion relative to each other. Therefore, this method is ineffective for measuring the distance between stationary transmitters and receivers.

2.3 Collision Avoidance with Obstacles

One of the most challenging aspects of routing is finding a suitable path for robots to navigate through unknown and dynamic environments [16]. This difficulty arises from the presence of dynamic constraints that make it increasingly complex for robots to avoid collisions while following their intended paths. To address this issue, various methods have been employed, including visual graph techniques, Voronoi diagram-based cell decomposition, virtual potential fields, and similar algorithms. Many of these methods rely on geometric approaches implemented within the configuration space. However, most of these methods encounter challenges such as getting trapped in local minima, inability to adapt to unknown environments in real-time, requiring significant freedom of movement to avoid obstacles, and high computational requirements [17].

To overcome some of these limitations, model-based predictive controllers and their variants, such as predictive horizon controllers, have been widely used [18]. Additionally, shape algorithms and enhanced shape models have been employed to tackle the mentioned issues and improve collision avoidance capabilities [19]. These approaches offer promising solutions in the field of collision avoidance in routing systems.

2.3.1 Bug Algorithm

The first algorithm proposed for obstacle avoidance is the Bug algorithm, which falls under the category of shape algorithms. This algorithm offers a simplistic approach to obstacle detection and avoidance. Its main concept involves circumnavigating detected obstacles in the environment. By completely bypassing the obstacle, the robot then proceeds towards its intended target, taking the shortest path available [20].

While the Bug algorithm ensures that robots in the network can reach all reachable targets, it is considered highly inefficient. The algorithm heavily relies on the current sensor readings to

navigate around obstacles. One drawback of this approach is that the robot cannot effectively navigate in close proximity to obstacles, resulting in longer task completion times. Consequently, the robot may not be able to accomplish all assigned tasks within the limited time available, despite its high accuracy.

2.3.2 Improved Shape Algorithm

The improved shape algorithm enhances the obstacle avoidance process by utilizing a more efficient path. In this algorithm, the robot initially moves along a straight line from the starting point to the target point. If an obstacle is detected along this path, the robot adjusts its trajectory to bypass the obstacle while staying on a line between the starting and target points. Once past the obstacle, the robot resumes its original path, making necessary adjustments to ensure a smooth transition. By doing so, the improved shape algorithm selects a significantly shorter path for the robot's movement [19].

Angle-based path planning methods play a crucial role in finding shorter paths by propagating information along the grid edges without confining the path exclusively to the grid edges. Network-based methods often involve continuous searching, especially when the robot's knowledge of the configuration space or the configuration space itself changes during path finding. Heuristic incremental search algorithms leverage the experience gained from previously solved path planning problems to rapidly generate optimal paths for the current scenario [21].

2.4 Algorithms Based on Sampling

Algorithms based on sampling provide an effective solution to the challenge of getting trapped in local minima and offer relatively fast problem-solving capabilities. These algorithms

employ sampling techniques that can assess the probability of failure in finding a feasible path. As time progresses, the probability of failure decreases, eventually converging to zero. Presently, sampling-based algorithms [22] are considered the most advanced and efficient motion planning algorithms, particularly in high-dimensional spaces. They can be successfully applied to a wide range of problems, including mobile robots, robotic arms, biomolecules, digital character animation, and many others. These algorithms excel in addressing problems with multiple dimensions, enabling their implementation even in scenarios with tens or hundreds of dimensions.

2.5 Distance Calculation Algorithm

The distance calculation algorithm is designed to determine the distance between two points in a data transmission network. The algorithm operates in the transverse layer of the network and involves the exchange of digital pulse strings between a robot transmitter and a target receiver.

The process begins with the robot transmitter sending a digital pulse string. Upon receiving the pulse string, the target receiver identifies the transmitter and responds by sending a specific pulse string of zero bits. The number of pulses transmitted from the robot to the target serves as the basis for calculating the distance between the two points [23].

To calculate the distance, a counter (I) keeps track of the number of pulses sent. When the target receiver receives the first bit of data, it sends a message instructing the robot transmitter to stop generating pulses. During the time it takes for the stop command to reach the robot receiver, the counter (I) doubles. Therefore, a factor of 1.2 is used to account for this doubling effect in distanceS calculation [15].

To estimate the distance, the number of generated pulses is multiplied by the length of each pulse (λ), which is a predetermined fixed value. The length of each pulse depends on the working frequency of the transmitter-receiver system used. The relationship $C/f=\lambda$, where C is the speed of

light and f is the frequency of the communication module, can be used to calculate the pulse length. Higher working frequencies result in shorter pulse lengths and improved distance calculation accuracy.

For accurate operation, it is essential to account for potential system errors, such as the production of additional pulses due to disparities between the processor speed and the speed of radio waves. These considerations are incorporated into the network distance calculation algorithm to ensure reliable and precise distance estimation [24].

2.6 Spiral Algorithm – Determining the Coordinates of the Target Point on the Path

During the implementation of any measurement system, operational errors arise, often influenced by atmospheric parameters such as pressure, humidity, and temperature. When calculating close distances and faced with the limitations of highly accurate calculation systems, the spiral movement algorithm in space is introduced to obtain necessary information for characterizing a point or reducing the distance to the target with minimal error approximation.

To address the challenges of calculating close distances and minimize the error in distance calculations, the spiral movement algorithm for robot flight utilizes the equations of Archimedes spiral [25] and exponential spiral [26]. This algorithm, based on fundamental geometric concepts in 3D space, equips the robot's flight system with the necessary information to determine the coordinates of a stationary target or narrow down the geometric location of possible target points.

Assuming an error of approximately $\pm \delta$ in distance calculations between two objects, the pulse count data used to estimate the distance from the robot to the target is stored in variable D (representing the distance obtained from the initial measurement). The actual distance falls within

the range [D - δ (m), D + δ (m)], and the target coordinates are derived within the geometric location of a sphere S with a radius of δ meters and center O.

Instead of following a straight path, the robot navigates a path resembling a spiral, as depicted in Figure 4. Based on the direction of progress towards the target, a linear equation is established for each step. The spatial angle of these lines is determined only after the target orientation is known. In other words, the robot can determine the coordinates of the target only after determining its direction. By traversing multiple steps, the robot identifies the target's location by intersecting these line equations. In the worst-case scenario, this algorithm yields a sphere with a radius smaller than δ, surrounded by intersecting lines, thereby achieving highly accurate determination of target coordinates with negligible error.

2.6.1 Exponential Spiral

The exponential spiral equation is given by:

r = ae^(-t)cos(t)i + ae^(-t)sin(t)j (1-2)

The velocity vector is:

V = ae^(-t)(-(cos(t) + sin(t))i - (sin(t) - cos(t))j) (2-2)

The magnitude of the velocity vector is:

|V| = √2 ae^(-1) ds/dt (3-2)

The unit tangent vector is:

T(t) = √(1/2) (-(cos(t) + sin(t))i - (sin(t) - cos(t))j)Ţ (4-2)

Therefore, the radius of curvature becomes:

ρ(t) = √2 ae^(-1) (5-2)

The derivative of the unit tangent vector with respect to arc length is constant:

dŢ/ds = k (6-2)

Since $N\hat{} = \rho(d\mathbf{T}/ds)$, the center of curvature, which is essentially the target location, can be expressed as:

$$rC(t) = r(t) + \rho(t)N\hat{}(t) = ae^{(-t)}(\cos(t)i + \sin(t)j) + 2a^2e^{(-2t)}(1/\sqrt{2}\, ae^{(-t)}((\sin(t) - \cos(t))i - (\cos(t) + \sin(t))j) = ae^{(-t)}(\sin(t)i - \cos(t)j) = ae^{(-t)}(\cos(t - \pi/2)i + \sin(t - \pi/2)j) \quad (7\text{-}2)$$

To follow a path aligned with the exponential spirals, the change in angular positions between points should be exchanged using angle sensors such as the L3G4200D. The robot repeats these steps until it reaches the target. It is worth mentioning that the robot can approach the destination directly for a certain length of the path and then switch to the spiral algorithm to enhance accuracy.

Based on the calculations [27] , in each step of acquiring information, the robot determines the precise location and coordinates of the target by applying geometric concepts and obtaining line equations from at least three line segments. The accuracy is increased by using the coordinates of the robot and the obtained coordinates of the target and aligning them with the direction of the target.

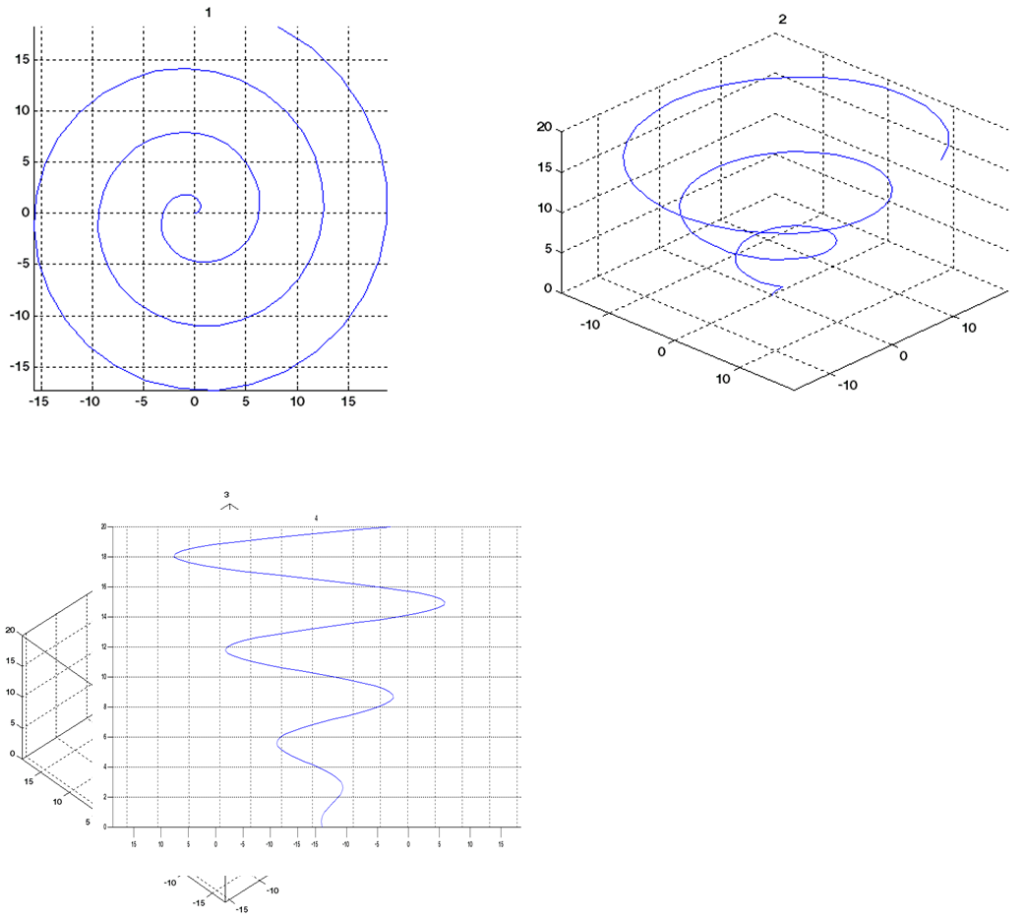
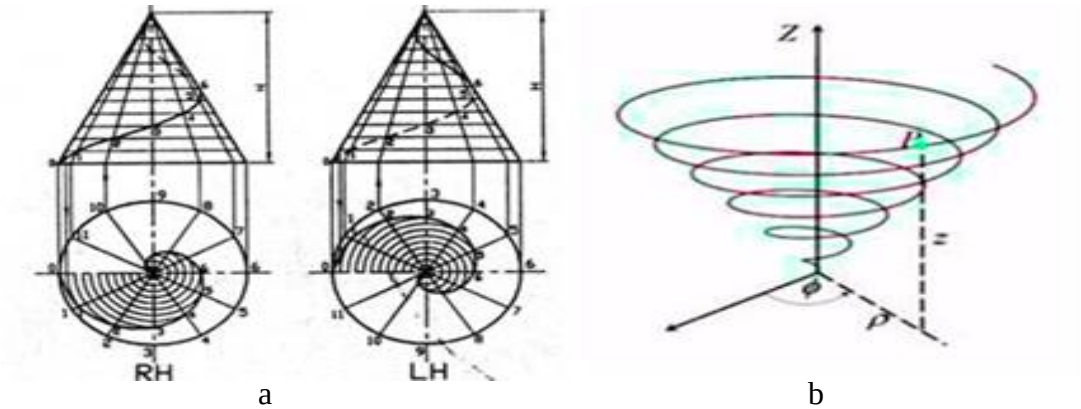


Figure 2.1. Spiral motion towards destination on a coordinate plane quadrant.

Spiral motion towards a destination on a quadrant of the coordinate plane entails moving along a curved path that progressively draws closer to the target while constantly revolving around it.
1. In this Figure 1, this concept is illustrated as a clockwise (right-handed) spiral motion within the coordinate system's quadrant. The X-axis ranges from -15 to 15, and the Y-axis similarly ranges from -15 to 15.
2. This three-dimensional space is depicted, showcasing a spiral trajectory. The spiral motion unfolds within a coordinate system defined by Y ranging from 0 to 20, X from 0 to 10, and Z from -10 to 10. The movement retains a right-handed (clockwise) rotation as it advances.
3. These spirals have been coordinated and aligned along a straight line to efficiently converge towards the intended destination point.

2.6.2 Archimedes Spiral (r=θ)

Based on the Archimedes Spiral equation (r=θ), the trailing object follows a path as depicted below. The direction of the spiral, whether it is left or right, depends on the environmental conditions and the presence of obstacles.



a b

Figure 2.2. Trajectory of the trailing object as a conical surface

Figure b: visually presents the trajectory of a trailing object as a conical surface, defined by key parameters: y as the horizontal component, Z as the vertical component, θ as the launch angle from the horizontal, P as the constant acceleration due to gravity.
Figure a: shows the direction of the spiral can be LH "Left Hand," or RH, "Right Hand.

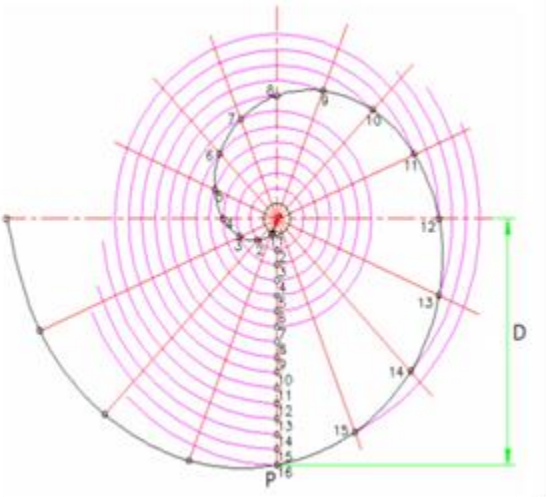


Figure 2.3. Top view of tracer movement using Archimedes' spiral to reach the target location.

This figure illustrates tracer movement using Archimedes' spiral toward a target. By calculating distances at waypoints 1-16 and employing linear equations, the method refines the target's location within a sphere. A wireless module with variable d adjusts the pace for accuracy, where lower d values enhance precision. In this instance, 16 distance calculations result in d=1/16 D advancement. This approach offers a precise method for target tracking.*

Starting from point P, the tracking object calculates the distance each time it reaches points 1, 2, ..., 16. Based on the direction of progress towards the goal, a linear equation is determined

for each step. By traversing multiple steps, the tracking object determines the location of the goal by intersecting these linear equations. In the worst-case scenario, this algorithm yields a sphere with a radius smaller than δ, surrounded by intersecting lines, narrowing down the geometric location of the possible target points. The wireless communication module board allows for adjustment of the progress rate in all stages using the variable d, which corresponds to the required accuracy (Burke & , 2005).

By decreasing the value of variable d, the number of samples increases, leading to higher accuracy of the system.

In Figure 2-3, the distance is calculated 16 times using the spiral algorithm, resulting in the progress amount of each step being equal to d=1/16 D*. Consequently, the tracker advances d meters towards the target with each step. The number of steps can be adjusted and calculated using the relationship n=D/d.

2.7 Network of air and ground robots

An intelligent mobile robot is required to reach predetermined goals within a specified time frame. At each step, the robot must accurately determine its location relative to the goals and employ suitable strategies to achieve them. Obtaining environmental information is essential for obstacle avoidance and optimal route planning. It is noteworthy that the objectives of implementing routing algorithms, path orientation, and obstacle detection include improving existing routing algorithms and comparing the performance and efficiency of different approaches.

Locating and identifying the optimal path from the starting point to the desired destination is a crucial challenge in numerous scientific and technological domains, particularly in the air and ground industries. This is especially relevant for the intelligentization and autonomy of rescue equipment, such as rescue robots and drones. Various technologies and systems have been

employed to accomplish this goal, including radar [28], LIDAR [15], GPS (Gupta, Morris , Patel, & Tan, 2012), and telecommunication satellites. Each of these systems is used in specific contexts for wave transmission purposes. However, despite the use of radar and GPS systems for object localization and mapping, they are considered ineffective for calculating distances between objects in the presence of obstacles.

Considering the potential limitations and practical issues associated with existing technologies and methods, there is a need for a new solution that eliminates reliance on advanced equipment and addresses these challenges. The system proposed in this article overcomes these limitations by enhancing accuracy, reducing costs, and enabling determination of coordinates for the origin and destination points in three-dimensional space. It achieves orientation solely based on information from the origin and destination stations, determines the direction of progress, tracks secondary objects, finds the optimal route to the destination, and calculates the distance between two stations. Furthermore, this system can be implemented on fixed or mobile devices using innovative methods and programmed algorithms.

In recent years, unmanned aerial vehicles have gained significant prominence in various fields. These robots are employed in scenarios where reaching the desired location is challenging. They offer advantages such as adaptability to different climatic conditions, high altitude capabilities, precise measurement of desired parameters, and rapid problem-solving abilities. Their flexibility and reliability make them effective solutions for a wide range of applications.

2.7.1 Creating robotic networks for data mining

Aerial or ground robot networks can be particularly valuable and effective in challenging conditions and environments where human intervention is limited. In such scenarios, these networks can be deployed for data mining and collection purposes, followed by data processing.

The following situations illustrate the applications of aerial or ground robot networks, or a combination of both, in this context. The primary objective of these networks is to enhance orientation and positioning through the implementation of movement decision strategies by the robots.

2.7.2 Functional description of robot networks in different fields

Given the extensive range of applications for programmed robots and the diverse array of human needs that can be addressed through their utilization, the following two scenarios highlight the use of aerial and ground robot networks. These networks are known to be highly efficient in various fields such as communication, transportation, and more.

2.7.3 Management of natural resources with the help of a network of robots

The first scenario involves the utilization of a robotic network for the management of natural resources, specifically in the context of a forest area affected by a high-risk fire.

In the overall management of natural resources, it is crucial to monitor the health of the resources, including the status of trees (whether alive or dead) and the height of vegetation in a given area. In many cases, forest areas are located in close proximity to power transmission lines, and excessive tree growth can lead to contact between the vegetation and the wires. This contact, caused by changes in tree diameter due to expansion and contraction, increases the risk of fire outbreaks.

To address this issue, a network of robots can be deployed in two phases to effectively manage the situation. The first phase involves using robots equipped with imaging capabilities to calculate the distance between trees and transmission lines during flight. This calculation is performed using mathematical rules such as Thales' theorem and line equations to determine if the

distance is below a certain threshold deemed dangerous. If the distance is critical, relevant authorities can be alerted to take necessary actions such as trimming the vegetation in the area, thereby reducing the risk of fire.

The second phase focuses on utilizing the robotic network in the event of a fire outbreak to locate the origin of the incident. In this scenario, robots equipped with sensors fly over the affected area. They use temperature measurements and a model dependent on those measurements to estimate the distance to the main fire area. This estimation is further refined based on previous information collected from the region. Through this iterative process, the robots update their position and ultimately perform the identification operation to pinpoint the source of the fire.

2.7.4 Multi-step data transmission in the robot network

Another scenario discussed here involves the development of a control scheme for robot movement within a covered area using a robotic network. This control scheme utilizes the distance between flying robots and the strength of their signal transmissions to establish a strong network connection and enable high-rate data transfer. Each robot within the network sends its data to both the ground station and the guiding robot, ensuring robust data transmission and processing.

The data transmitted includes information from motion detection sensors embedded in both ground and aerial robots. These sensors may include accelerometers, gyroscopes, GPS, barometers, compasses, and more. The output data from each robot serves as input data for other robots within the network, allowing for data processing and analysis by multiple robots.

For instance, in a scenario where robots are following a straight path based on a line equation, each robot needs to receive commands to detect the continuation of the path and transmit the obtained data to other receiving robots. This multi-step data transmission ensures that the data

remains intact and can be further processed by subsequent robots in the network, contributing to efficient and collaborative movement control.

2.7.5 Network-based algorithms

Network-based algorithms are capable of tackling highly intricate, continuous, and diverse problems. These algorithms provide solutions by leveraging network visualization over the configuration space. One such challenging problem that can be addressed by these algorithms is precise motion planning for systems with higher dimensions and intricate constraints, all while navigating complex obstacles. However, due to the computational complexity involved, achieving efficient solutions for these problems at the local level is often challenging.

2.7.6 Network search

Grid-based algorithms typically superimpose a grid on the configuration space and assume that each configuration is defined by a point of the grid. At each point of this network, the robot is allowed to move to the adjacent point so that the line between the start and end points inside Cfree or the same space of this problem is tested using collision detection. The set of these actions together with the search algorithms can find a path from the origin to the goal. These methods require network separation. The search becomes faster with thicker grids, but in this case the algorithm fails to find a path through the narrow Cfree sections. In addition, the number of points on the grid grows exponentially in the dimensions of the configuration space, which makes them inefficient for high-dimensional problems.

Traditional network-based methods produce trajectories whose direction of path changes is limited to randomness

It is from the given initial angle. These methods often result in paths that are partially

are optimal.

2.7.7 The performance evaluation of a robot within a network

The performance evaluation of existing systems that rely on two stations for navigation is primarily based on the strength or timing of the transmitted signal [29]. However, in certain scenarios, image processing plays a crucial role, although its practicality may be limited in certain fields, and it can significantly reduce measurement accuracy [30].

The presented system introduces a novel approach to orientation during the tracking process. Simply calculating the distance (D) alone does not provide sufficient information to determine the target's direction. To address this, the system utilizes geometric calculations to establish a sphere with a radius of D centered at the current location of the tracking object. By employing spiral or straight movement techniques and leveraging mathematical knowledge, the follower explores all eight quadrants of the coordinate axis, along with their respective halves, to identify the correct quadrant that leads towards the target station. As the follower continues its movement, the orientation becomes increasingly accurate, and upon achieving the desired precision, the built-in target coordinate identifiers relative to the starting point of movement are utilized.

In cases where the target is in motion, after several steps have been taken, the changes in sensor readings between the two objects are sufficient for both direction and alignment. If the following object maintains an adequate speed, they will eventually come closer or maintain the desired distance within the desired range. All these settings can be adjusted by the user.

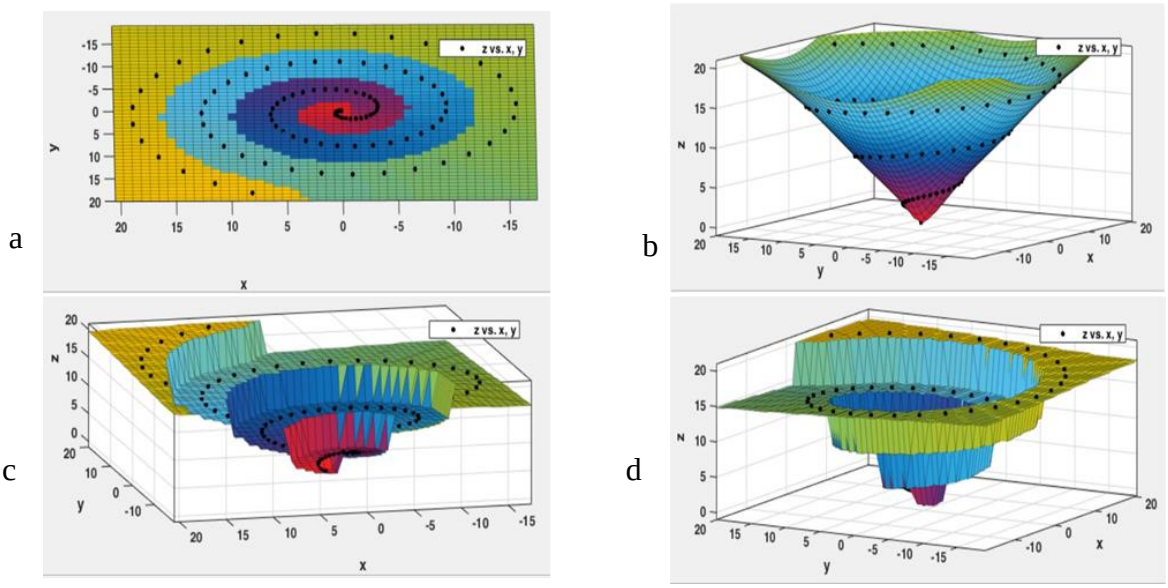


Figure 2.4. Trajectory of the Tracking Robot

a: The first diagram (Figure 2.4a) demonstrates the initial movement of the tracking robot in a two-dimensional space. The X-axis spans from 20 to -15, and the Y-axis ranges from 20 to -15. The motion follows a left-handed (counter-clockwise) direction.
b: Figure 2.4b presents a three-dimensional view of the robot's trajectory. This diagram depicts the robot's motion in a multi-dimensional space defined by N (0-20), Y (-15 to 15), and X (-10 to 20).
c: In Figure 2.4c, a distinct perspective of the 3D trajectory is shown from one angle. The spirals converge toward the center point (0, 0), illustrating the coordinated movement to reach the origin.
d: Finally, Figure 2.4d provides another viewpoint of the 3D trajectory, showcasing how all movements align and merge towards the central point (0, 0). This alignment highlights the efficient strategy employed by the tracking robot to converge on the target point.

2.7.8 Aerial robot control system

In the design of a high-performance intelligent system, adherence to mechanical and aerodynamic principles and rules is more crucial than the design of its electronic board. Despite the implementation of precise PID controllers [31], unfavorable weather conditions and external forces during flight can momentarily disrupt the robot's balance and potentially lead to a fall. To address this, the presented system adopts an X-shaped design model, which offers a more resilient and lightweight body compared to a complete square model. The continuous change in movement

direction necessitates symmetrical weight distribution in the robot. To achieve this, motor protectors have been designed to adjust the robot's weight in different movement directions.

To maintain balance and facilitate communication between two flight phases (automatic routing and aiming, and tracking obstacles and secondary objects), an urgent need arises for controller design. The system controller is designed to collect feedback from both phases and combine their outputs. It employs an intelligent control system consisting of P-I and D terms. This control system takes into account and automatically corrects past and present errors while predicting potential future errors and the structure's position. The controller provides voltage outputs to the built-in speed controllers, which adjust the motor direction based on these changes.

The controller comprises a P-I term for vertical direction error correction with spark control and another controller consisting of P-D terms for horizontal direction error correction to account for path deviation. These two controllers are combined to form an optimal PID controller, aiming to increase profit and enhance decision-making accuracy in state identification. After object identification as the first step, the system automatically adjusts the PITCH angle in the movement direction using the Marpage algorithm. It takes several samples, applies an appropriate filter to reduce image noise, and processes the data to obtain the desired result.

2.7.9 Evaluating the Performance of Multiple Robots in a Network

Using multiple ground or aerial robots simultaneously to cover a region offers an alternative approach compared to using a single robot. In such a network, information can be exchanged between the robots, enabling communication among network members. Ensuring a secure and stable connection within the network is a crucial aspect that requires a suitable solution. The primary challenge in this network is to collect data from the surrounding environment and guide it without alterations until it reaches a base or reference station.

The FANNet network topology comprises different aerial robots, along with a guiding or controlling aerial robot responsible for collecting data from other robots and transmitting it to the ground station. A model has been proposed to locate the guiding robot among other robots at any given moment. The objective of this model is to enable the guiding robot to fly to an optimal point based on the desired position and final goal. The ideal point represents a suitable position for the robot in terms of communication with other devices in the network. Estimating this positioning relies on the positions of other robots in the network, followed by calculating and optimizing the ideal point within the range of the guiding robot.

The movement of these robots in the network is divided into two independent phases, involving remotely controlled flying or ground robots. The simultaneous performance of multiple flying robots in the network has been examined. The position, traffic, and movement of other flying robots in relation to the guiding robot are utilized to train the network, ultimately determining the position of the guiding robot or the reference point within the network space.

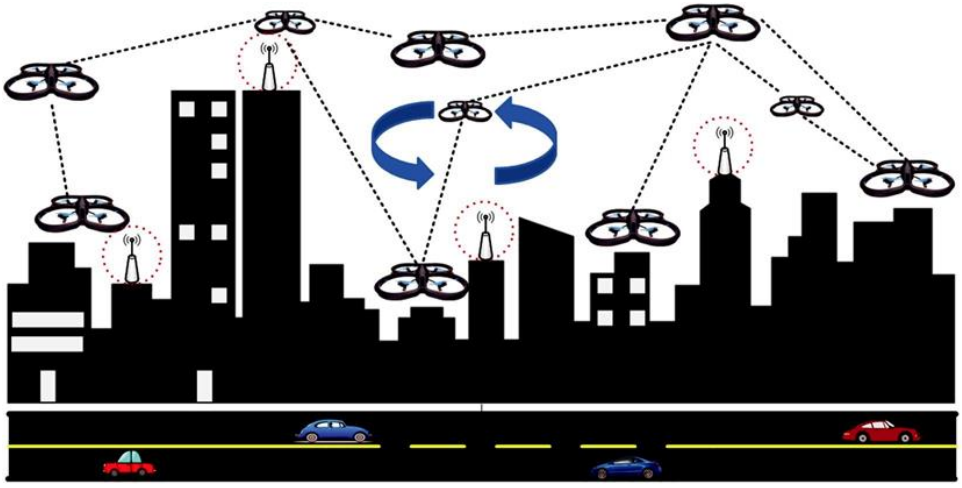


Figure 2.5. Aerial Robot Network

The aerial and ground robots establish communication facilitated by a WiFi node, enhancing coverage for their interaction.

2.7.10 Network architecture of autonomous robots

Network architecture of autonomous robots aims to establish an autonomous robot network (United States Patent No. 6,266,577., 2001). To achieve this, all robots utilize their inherent positioning capabilities to determine relative and absolute distances, detect obstacles along their path, and identify network robots and nodes that serve the main purpose.

The process begins by detecting the current position of each robot. Subsequently, the motion data flow from the flying robots is captured, and their current positions are reevaluated. If the time and distance fall within the allowable range, the node is designated as the position of the guiding robot or reference. The new position of the guide is promptly communicated to the network. If the time and distance exceed the acceptable thresholds, the robot returns to its starting point. This iterative process continues until the desired and optimal point is identified, allowing for effective coordination and achievement of collective goals within the autonomous robot network (United States Patent No. 6,266,577., 2001).

2.7.11 Features of the FANNet network

One of the notable features of the FANNet network [32] is its decentralized system components, where each node operates independently without following a specific infrastructure. This approach allows nodes that can establish connections with other components to act as paths or guides, prioritizing communication. These nodes have the flexibility to move arbitrarily and alter the network's topology, necessitating constant compatibility and route reconfiguration to maintain inter-node communication.

In comparison to other traditional networks like "AdHoc" [33], FNET network exhibits a higher mobility index, resulting in frequent topological changes that enhance area coverage. Essentially, the FANNet network is a self-adjusting network and an autonomous planner, capable

of promptly adapting to sudden changes in topology, node organization, and communication systems.

Continuous monitoring of mobile unit mobility and spatial arrangement is crucial to determine communication routes. To maximize node autonomy and minimize data transmission delays from sender to destination nodes, dynamic routing is employed, emphasizing effectiveness, efficiency, and simplicity. Communication between robots in the network relies primarily on their locations. The guiding robot node centrally collects environmental information and redistributes data to the control station. If the guiding robot has strong connections with certain network robots while maintaining weak connections with others, the reference node or guide may not be optimally positioned.

Flying robots play a significant role in the FANNet network as they navigate the environment and collect data using sensors. Notably, the planned FANNet network has a high data transmission capacity and the ability to capture and monitor environmental conditions. Each device within the network must be aware of the locations of other network components at all times. Therefore, the use of GPS, which provides location-related information at an average rate of one update per second, is necessary. Additionally, the robots are equipped with an internal measurement unit (IMU) that allows them to transmit their positions at any moment within a shorter timeframe compared to GPS updates.

The settings of devices used to cover an area directly impact network efficiency and can either enhance or weaken the network based on their mobility. This represents one of the critical challenges addressed by this type of network.

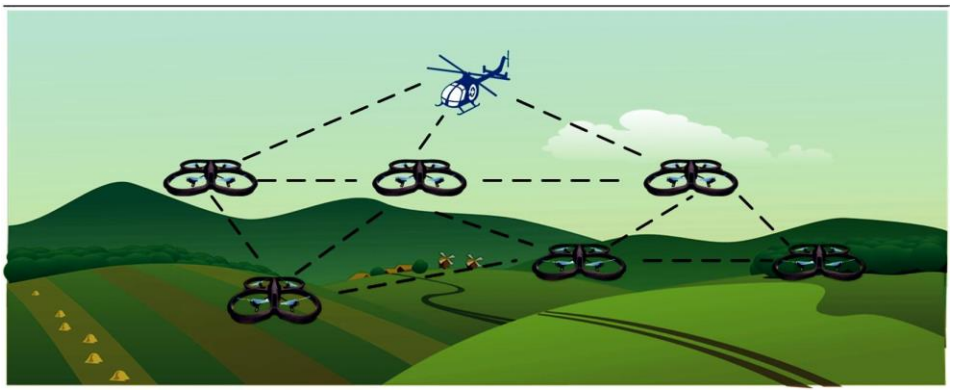


Figure 2.6. FANNet network scenario

In this illustrative depiction, we peer into the network scenario of the innovative FANNet (Future Air-Ground Network) system. A harmonious fusion of aerial and ground components unfolds as they communicate seamlessly.

3 Research and Experiment Methodology

This chapter presents the research methodology employed in this study to investigate the performance of the Bat Algorithm and Genetic Algorithm in solving optimization problems. The research methodology serves as a roadmap for conducting the research, outlining the overall design, approach, and procedures used to collect and analyze the data. It provides a framework to ensure that the research objectives are achieved, and the research questions are effectively addressed.

In this chapter, the various components of the research methodology will be discussed in detail. The chapter begins by explaining the research design, which serves as the foundation for the entire study. It outlines the overall structure and plan for conducting the research, including the selection of algorithms, benchmark functions, and performance metrics.

3.1 Research Approach

The research approach describes the overall strategy used to address the research problem. In this study, a quantitative research approach was adopted to collect and analyze numerical data. This approach facilitated the evaluation and comparison of the performance of the Bat Algorithm and Genetic Algorithm in solving optimization problems.

3.2 Sampling Technique

The sampling technique utilized in this research determines the selection of participants or data points for analysis. Since the study focused on optimization algorithms and their performance, a non-probabilistic purposive sampling technique was employed. This allowed the researcher to handpick specific algorithms and benchmark functions that represented the target population.

3.3 Data Collection

Data collection involved gathering relevant information and measurements to address the research questions. The primary sources of data for this study included benchmark functions and performance metrics obtained from simulation experiments. These data were collected through computer-based simulations and recorded for further analysis.

A test dataset is utilized to train the model based on GPS using Neo-6m. The latitude and longitude values within the data indicate the corresponding locations, and there is a target for evaluating the model. The test data can be found in the appendix.

3.4 Experimental Procedure (if applicable)

The experimental procedure outlines the steps followed to conduct the simulations and collect data. In this study, the Bat Algorithm and Genetic Algorithm were implemented using a programming language, and a series of experiments were conducted to evaluate their performance.

The benchmark functions were used as test cases, and the algorithms were run iteratively to obtain performance measurements.

3.5 Data Analysis

Data analysis refers to the process of transforming collected data into meaningful insights and drawing conclusions from the findings. In this study, the collected data were analyzed using statistical techniques such as performance comparison, mean analysis, and graphical representation. These analyses provided a quantitative evaluation of the algorithms' performance and allowed for meaningful comparisons to be made.

3.6 Limitations

One limitation of this thesis is the availability of data for training the algorithms. Due to constraints in data collection or access to relevant datasets, the research might have been limited by the amount and diversity of data used in training. Having a larger and more diverse dataset could have potentially improved the performance and accuracy of the algorithms. The availability of additional data could have allowed for more comprehensive training, enabling the algorithms to capture a wider range of patterns and variations in the data.

Furthermore, the limitations in data might have restricted the generalizability of the algorithms. With a larger dataset, it would have been possible to test the algorithms on various scenarios and evaluate their performance across different conditions. This would have provided more robust insights into the effectiveness and applicability of the trained algorithms.

It is important to acknowledge that the quality and quantity of data play a significant role in the training and performance of machine learning algorithms. Having more data available for training would have allowed for a more thorough exploration of the algorithm's capabilities and potential improvements in its overall performance.

4 Result and Finding

In previous systems, a ground station was used to process wirelessly transmitted data. The data underwent processing at this station and was then returned to the system through a network. Consequently, this delay caused disruptions in the execution process and hindered access to corrected data required for determining the route's continuation. This is due to the constant possibility of various forms of communication, such as pathfinders, satellites, radar, LIDARs, and others being available at any given moment. Additionally, these systems were prone to errors in accurately detecting the coordinates of points, resulting in imbalances and reduced visibility of the road ahead. Moreover, the associated costs of equipment and the need for manpower further increased. In contrast, the system being presented achieves processing with higher accuracy at a reduced cost.

4.1 The Importance of Utilizing an Optimal Routing Algorithm in the Network

The utilization of an optimal routing algorithm in a network is crucial for the maneuverability of any robot operating within it. Consequently, within the entire system of each robot, group behavior should exhibit adaptability and flexibility based on the complexity parameters of the movement path. Planning the tasks of a communicating flying robot [34] within the network involves numerous parameters and dynamic variables. Each algorithm is employed for a specific scenario, such as distributing an intelligent system to automatically determine coordinates, establish communication with other robots

via signal transmission, and enable decision-making and task execution in real-time. However, thus far, high-performance data transfer techniques for robot positioning have not been utilized. Consequently, all existing systems continue to operate despite the limitations in this aspect of movement, which commonly involves wireless communication and both on-board and remote calculations [35].

4.2 The Feasibility of Utilizing Optimization Algorithms for Routing

In order to optimize the movement of multiple flying robots, a designated robot is identified as the reference point for positioning other robots within the network. This optimization process ensures that the flying robot remains close to other network robots along its trajectory.

This optimization is made possible by the guiding robot, which obtains the precise position of each robot and shares this information with others in the network. Consequently, each robot is directed towards the estimated new position of the guiding robot. A visual representation in this section illustrates the impact of applying optimization versus not applying it on network traffic and stability. It is evident that without the relevant algorithms, the traffic between robots and the reference robot becomes more unstable, resulting in a time delay of approximately 0.051 milliseconds.

When network robots are at a greater distance from the guiding robot, despite having strong communication capabilities, their mobility increases as they navigate towards the guiding robot. The optimal outcome is achieved when the average time delay is around 0.03 milliseconds. Another image depicts a scenario in which routing is performed without optimization until reaching the target point. In such cases, estimating

the distance between the guiding robot and other robots becomes impossible. However, the optimization algorithm can calculate the new position of the guiding robot in real-time, assuming a strong connection between the network robots.

a

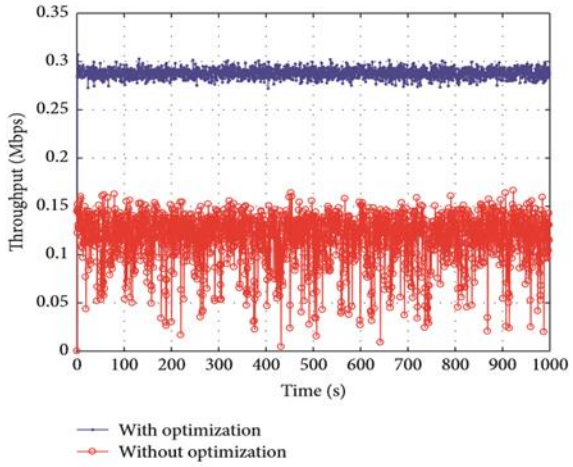


FIGURE 9: Throughput performance.

b

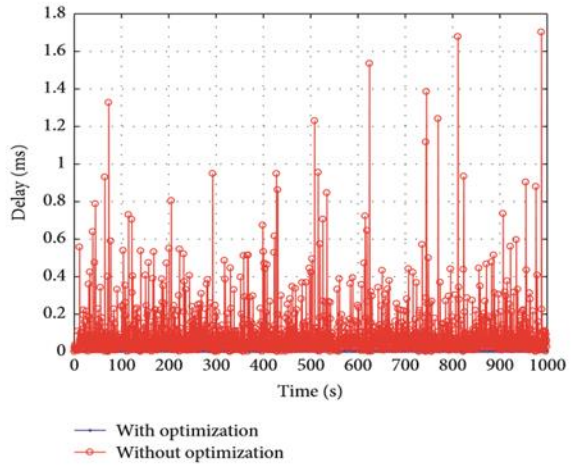


FIGURE 10: Delay performance.

Figure 4.1. Network Time Delay Comparison with and without Optimization

This figure dissects optimization's impact on throughput and delay metrics. Red (unoptimized) and blue (optimized) scenarios contrast along time (s) on the horizontal axis, while throughput (Mbps) and delay (ms) line the vertical axes.

a. Throughput optimization is evident in the blue curve, maintaining a consistent throughput rate of 0.3 Mbps. In stark contrast, the red curve exhibits erratic throughput ranging between 0 and 0.15

Mbps. This disparity underscores the effectiveness of optimization in stabilizing and enhancing throughput performance.

b. *Notably, delay values fluctuate extensively between 0 and 1.7 ms in the unoptimized red scenario. However, following optimization, delay values experience a remarkable reduction and near elimination. This tangible outcome accentuates the crucial role of optimization in significantly mitigating delay, leading to smoother and more efficient operations*.

4.3 Nature-Based Optimization Metaheuristic Algorithms

Nature serves as a rich source of inspiration, offering diverse and dynamic phenomena that can help solve complex problems across various scientific disciplines. In the field of optimization, where non-linear fitting of models and curves is crucial, researchers seek to find the optimal allocation of resources in society and industry. This has led to the development of metaheuristic algorithms [35] that draw inspiration from nature and employ intelligent techniques to solve optimization problems, revolutionizing computational methods.

Although significant research efforts have been devoted to this area over the past decade, the field of nature-based optimization algorithms is still relatively young, with ongoing surprising results. The exciting aspect of these algorithms lies in their ability to expand the scope and feasibility of applications, opening up new opportunities in computer science, calculations, industries, and other domains.

In essence, a metaheuristic optimization algorithm represents an innovative approach that can be applied to various optimization problems with minimal modifications. These algorithms greatly enhance the ability to find high-quality solutions for challenging optimization problems. One common characteristic of metaheuristic algorithms is their

utilization of mechanisms to escape from local optima. They can be broadly categorized into two groups: solution-based algorithms and population-based algorithms.

Solution-based algorithms modify a solution during the search process, focusing on local areas of the search space. On the other hand, population-based algorithms consider a population of solutions and explore different regions of the solution space simultaneously. These algorithms can be viewed as random search methods employed to find the optimal solution.

Furthermore, these algorithms can be further classified into exact algorithms and approximate algorithms. Exact algorithms are capable of finding the optimal solution precisely, but they suffer from scalability issues and exponentially increasing execution times for complex optimization problems. In contrast, approximate algorithms aim to find near-optimal solutions within a shorter time frame, making them suitable for hard optimization problems. Approximate algorithms are categorized into three groups: heuristic algorithms, metaheuristic algorithms, and super-heuristic algorithms.

Heuristic algorithms face challenges such as getting trapped in local optima and premature convergence. Metaheuristic algorithms have been developed to overcome these limitations associated with heuristic algorithms. They offer solutions for escaping local optima and can be applied to a wide range of problems. Over the past few decades, various categories of metaheuristic algorithms have emerged, representing subsets of the broader metaheuristic algorithm family.

Overall, nature-based optimization metaheuristic algorithms provide powerful tools for tackling complex optimization problems, leveraging the wisdom of nature to enhance computational techniques and drive progress in diverse fields.

4.3.1 Four Different Classification Criteria of Metaheuristic Algorithms

- Based on one answer and based on the population: Metaheuristic algorithms can be classified into single answer-based and population-based algorithms. Single answer-based algorithms modify one solution during the search process, while population-based algorithms consider a population of solutions. Well-known population-based metaheuristic algorithms include evolutionary algorithms, genetic algorithms, genetic programming, ant colony optimization, bee colony optimization, particle swarm optimization, championship algorithms in sports leagues, optimization inspired by light physics, root algorithm, and the smart water drop algorithm. On the other hand, single answer-based algorithms include the forbidden search algorithm and the simulated annealing algorithm.
- Inspired by nature and not inspired by nature: Many metaheuristic algorithms draw inspiration from nature, mimicking the behavior of living organisms. Examples of nature-inspired metaheuristic algorithms include the gray wolf optimization algorithm, dragonfly optimization algorithm, flower pollination optimization algorithm, whale optimization algorithm, grasshopper

optimization algorithm, and bat optimization algorithm. However, there are also metaheuristic algorithms that are not directly inspired by nature.

- Metaheuristic algorithms with memory and without memory: Some metaheuristic algorithms employ memory, storing and utilizing information obtained during the search process. Examples of memory-based metaheuristic algorithms include the forbidden search algorithm. On the other hand, some metaheuristic algorithms do not rely on memory and do not utilize information obtained during the search.
- Deterministic and probabilistic: Metaheuristic algorithms can be classified as either deterministic or probabilistic. Deterministic metaheuristic algorithms, such as the forbidden search algorithm, make decisions based on determinism. In contrast, probabilistic metaheuristic algorithms, like the simulated annealing algorithm, employ probabilistic rules during the search process.

Additionally, it is worth mentioning crowd intelligence or group intelligence, which is a type of artificial intelligence based on the collective behavior of decentralized and self-managed systems. These systems consist of a population of simple actors that interact locally with each other and their environment. Examples of crowd intelligence can be observed in nature, such as swarms of ants, birds, animals, bacteria, and fish.

4.4 Crowd Intelligence Applied in Network Search

Crowd intelligence, also known as group intelligence, is a form of artificial intelligence that relies on the collective behavior of decentralized and self-managed systems. These systems typically comprise a population of simple actors who interact

locally with one another and their environment. While there is typically no centralized control dictating the behavior of these actors, their local interactions give rise to overall emergent behavior. Examples of such systems can be observed in nature, such as swarms of ants, birds, animals, bacteria, and fish.

Group robotics is an application that applies the principles of group artificial intelligence to a large number of inexpensive robots. Capturing group intelligence finds utility in information technology, where similarities exist between various issues in the field and the behaviors of social insects. A distributed system of independent and interacting actors aims to optimize efficiency and power while relying on self-regulation as a means of control and decentralized cooperation. Work distribution and task assignment also occur in a distributed manner. These applied examples of crowd intelligence and group intelligence stem from indirect interactions.

Among the most significant current and future applications of crowd intelligence, the following benefits can be highlighted: network routing and optimal resource allocation. The details of network routing will be discussed in the subsequent section, focusing on how crowd intelligence facilitates the efficient allocation of resources.

4.5 Correspondence of Optimization Problems with the Phenomenon of Natural Selection

The utilization of synergism and the concept of a sub-brain has become increasingly prevalent in engineering solutions for scientific and practical problems. From a hardware perspective, inspiration can be drawn from observing a dragonfly's flight to develop helicopter flight techniques or from studying the blind flight of bats to synthesize, design,

and construct radar systems. Similarly, in the software sector, it is possible to leverage the correspondence and analogy found in nature to address various challenges. One such intelligent application involves designing and solving engineering optimization problems by drawing parallels with the process of natural selection.

Typically, when observing natural laws, two perspectives are envisioned and subsequently tested. The first involves hypothesizing about the purpose and origin of creation in order to unravel the secrets behind it. The second perspective involves generalizing these insights to other realms of thought through the establishment of analogies and conceptual correspondences between two related cognitive systems. For instance, the remarkable diversity and abundance of different organisms raise intriguing questions. Another perspective involves recognizing the varying levels of complexity within individuals of plant and animal communities. In simpler terms, the current situation can be understood as the result of repetitions, interactions, and conflicts among various known and unknown forces, akin to a battle for survival.

It is well understood that the objective function represents a form of individual and/or collective selfishness, driven by the desire to survive. Those who exhibit greater grace, adaptability, and strength are more likely to thrive. Quantifying the degree of fitness, which is a fundamental aspect of modern engineering sciences, is a subsequent matter that will be addressed later. At this stage, we are focused on establishing a mental framework and a foundational correspondence. The process of evolution relies on an engine and a mechanism that make organisms more adaptable and compatible by approximating certain

characteristics. The environment itself, along with the frictions it presents and the constraints it imposes, mirrors the challenges encountered in optimization problems.

4.6 Permutation in Optimization

In optimization problems, various rearrangement operators that modify values are commonly employed. These operators are typically utilized when there is a need to rearrange a sequence in order to achieve a problem's solution. For instance, consider the traveling salesman problem, where the objective is to find the shortest closed route. In this problem, the salesman must visit all cities, with each city being visited only once. Assigning a number to each city allows us to represent a potential solution as a string. In this string, the city number indicates the visitation order, and the appearance or position of the city code signifies the sequence of city visits.

When applying rearrangement operators, any change made to the string represents a potential solution in which each city is visited exactly once. However, the truncation operator alone cannot guarantee an optimal rearrangement. As a result, an alternative operator known as the pseudo-dominant truncation operator is employed to increase the likelihood of the offspring string resembling its parent. To utilize this operator, two breakpoints are selected on the parent strings. Subsequently, the substrings between these breakpoints are exchanged with one another. Finally, the strings are recombined to preserve the modified values resulting from the exchange.

4.7 Selection and Comparison of Optimization Algorithms

In the realm of optimization, a wide array of algorithms exists, each possessing distinct strengths and weaknesses in terms of execution time, accuracy, and other relevant parameters. The task of choosing the most suitable optimization algorithm for solving a specific problem and evaluating its efficiency becomes paramount. An erroneous selection can introduce complex parameters that impede progress towards achieving the desired execution speed and result accuracy, or even lead to additional complications that deviate from the main objective. Thus, the process of selecting an algorithm for optimizing problem solutions necessitates meticulous consideration of the limitations and conditions associated with each algorithm.

Furthermore, it is noteworthy that the utilization of metaheuristic algorithms inspired by nature, while accounting for their unique characteristics, often yields resemblances to the problem conditions. This approach brings the optimization process closer to real-world scenarios and emulates natural phenomena. It is crucial to emphasize that the choice of a metaheuristic optimization algorithm for a given system should be based on specific reasons, taking into account factors such as the similarity of movement mechanisms and data transmission within the context of routing and data movement. For example, in scenarios involving networks of aerial and ground-based robotic systems, algorithms such as genetics or ant colony optimization can be considered.

4.8 Examination of Two Metaheuristic Algorithms

This project aims to leverage the strengths of existing systems while incorporating the benefits of optimization algorithms to achieve the desired network objective. The goal is to find an optimal path that avoids obstacles and facilitates efficient data transfer. Central and on-board processing are employed to effectively solve this problem, ensuring faster and more accurate travel from the origin point to the destination point. Two specific metaheuristic algorithms, namely the Bat Algorithm and the Genetic Algorithm, are examined in detail. These algorithms were selected from a pool of options, including the Ant Colony Optimization and Particle Swarm Optimization, based on specific criteria. The primary reason for their selection is the resemblance of their movement mechanisms and data exchange processes to those observed in ground and aerial robotic systems. Bats, for instance, follow a reference path during their movement. Considering the widely recognized and extensively used nature of the Genetic Algorithm as an optimization technique, comparing it with the metaheuristic optimization algorithm proposed by the Bat Algorithm can shed light on the distinctive features of the Bat Algorithm and highlight its superiority over other algorithms. This comparison further demonstrates the availability and applicability of the Bat Algorithm within the scope of this project.

4.8.1 Genetic Algorithms

Genetic algorithms are widely employed as metaheuristic methods for solving combined optimization problems. These algorithms simulate the process of gradual evolution found in nature by incorporating various patterns such as selection, crossover, mutation, and replacement. Each of these patterns is associated with specific operators

whose behavior is influenced by parameters like crossover probability and mutation probability. It is important to note that different combinations of these patterns and parameters yield different outcomes in terms of solution quality and convergence time. Therefore, the efficiency of a genetic algorithm depends on the careful design of operators, selection of the optimal combination among them, and finding suitable parameter values.

This report first presents common patterns used in genetic algorithms and then introduces an approach based on experimental design and response surface methodology. The goal is to identify the optimal combination of patterns and determine the optimal functional points for algorithm parameters.

In the genetic algorithm, inspired by the concept of gradual evolution, variables of the problem are encoded using established methods. Each encoded string represents a set of variables known as a chromosome. The fitness of a chromosome is evaluated based on an objective function. The algorithm begins with an initial population of chromosomes as the first generation and proceeds by generating new generations through the combination of individuals from previous generations. The parents, selected through specific methods, undergo crossover and mutation operations to produce new offspring. The new generation is formed by selecting the fittest offspring using predefined selection mechanisms. This iterative process continues until the termination criteria are met, indicating the completion of the genetic algorithm's evolution.

While numerous studies have focused on parameter tuning in genetic algorithm optimization, the construction of algorithm components and the selection of the most effective strategies have received comparatively less attention. This article aims to address

this gap. To design an efficient genetic algorithm, three steps are proposed: (1) developing diverse strategies for constructing the algorithm, (2) determining the optimal combination among these strategies, and (3) fine-tuning the parameters of the optimal strategy combination during the optimization process. In cases where multiple factors affect the studied systems, factorial designs offer comprehensive insight into the effects of these factors on algorithm efficiency. Therefore, employing factorial designs in the design of genetic algorithms is recommended to understand the impact of different patterns on algorithm performance. Additionally, statistical tests such as the least significant difference test or the multi-domain test can be utilized to determine significant differences among factor averages.

4.8.1.1 4.8.1. Hybrid Approaches in Optimization

In practical scenarios, certain optimization problems require a combination of different methods to achieve the optimal solution [36]. One common approach involves combining two genetic algorithms with a classical search method. This method is founded on the notion that while a classical search method may possess strong search capabilities, it can be sensitive to the initial guess. By integrating genetic algorithms into the optimization process, we can utilize their ability to explore the vicinity of local extremum regions. Subsequently, we can transition to the classical method to facilitate a faster and more accurate search.

It is important to note that the term "hybrid" is also used to describe methods that deviate from the traditional genetic algorithms introduced by Holland. Consequently,

certain operators such as jump or termination may need to be modified accordingly to suit the hybrid approach.

4.8.2 Bat Algorithm

The bat metaheuristic algorithm draws inspiration from the natural behaviors of bats [37]. With 996 known species, bats exhibit remarkable abilities such as echolocation, keen senses of sight and smell, and a diverse range of sizes, ranging from tiny bee bats weighing only 1.5 to 2 grams to large bats with a wingspan of 2 meters and weighing 1 kg.

Of particular importance is the way bats navigate and search for prey using sonar. By employing reflexive behaviors, bats can detect prey and avoid obstacles during flight. Throughout their flight paths, bats adapt various parameters, including changes in speed and location, in response to factors such as target identification, hunting strategy, and Doppler effect-induced variations in prey movement.

Bats' reflexive behavior enables them to identify prey while simultaneously avoiding obstacles and following dynamic paths to reach their destination. They emit short FM signals and pulses with a constant frequency ranging from 25 to 150 kHz. Each pulse has a duration of approximately 8 to 10 milliseconds, allowing for rapid assessment of the environment. When detecting obstacles along their path, bats emit ultrasonic pulses that last around 5 to 20 milliseconds. Notably, small bats or micro bats emit these pulses 10 to 20 times per second.

4.8.2.1 Ultrasonic Pulse Emission by Bats

Bats utilize ultrasonic pulses to detect obstacles in their flight path. When hunting prey in close proximity, bats can emit these pulses at a rate of approximately 200 pulses per second. Each pulse provides the bat with a significant amount of signal processing information. The sound emitted by bats has a propagation time of approximately 300 to 400 microseconds. With a power of 110 decibels, the ultrasonic waves produced by bats fall within the frequency range determined by the speed of sound in air (approximately 340 m/s) and the constant frequency emitted by the bat. Consequently, the emitted sound has a specific wavelength.

4.8.2.2 Bat Society

Bats, like other creatures in nature, have an inherent drive to seek optimal positions. As they navigate through their environment, they constantly strive to reach new positions. One crucial factor in achieving this is their awareness of their previous position and the position of the best bat within their bat community or network [38]. Just like in any society, bats in flight emulate the behaviors of the best bat and move towards it. Bats that are fortunate enough to receive the signals or pulses from the best bat can adjust their movement accordingly. This is because there may be other advantageous points located near the position of the best bat. The goal is to navigate through the space between each bat's initial position and the position of the best bat.

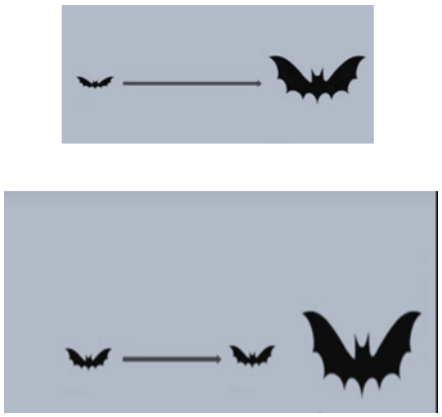


Figure 4.2. Movement Visualization between Initial and Best Bat Positions

In this visualization, the bat at the pinnacle, represented as larger, reigns as the winner with the best position. Other bats tirelessly strive to ascend towards this champion point, illuminating the dynamic pursuit of optimal positioning.

4.8.2.3 The characteristics of each bat

The characteristics of each bat in the bat algorithm are determined by three key parameters: position "X", velocity "V", and the quality or value of the objective function. These parameters are essential for the proper functioning of the entire movement network. Incomplete or inaccurate information regarding any of these parameters can lead to disruptions in routing and communication within the network.

To update the position and velocity of each bat, the following equations are used:

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta,$$

$$\mathbf{v}_i^t = \mathbf{v}_i^{t-1} + (\mathbf{x}_i^t - \mathbf{x}_*) f_i,$$

$$\mathbf{x}_i^t = \mathbf{x}_i^{t-1} + \mathbf{v}_i^t,$$

4.8.2.4 Description of the Bat Algorithm

The bat algorithm is inspired by the flight behavior of bats in nature, where a bat serves as a reference or guide for other bats to follow towards a desired destination. To describe the steps of this algorithm, the following procedures are performed [37] (Wang & Lihong , 2013).

Step 1: Define the objective function and initial parameter values, such as pulse rate, emission sound frequency, and related sound parameters.

- Set the pulse rate (r), which can be either constant or decrease during the algorithm.
- Define the upper (QMAX) and lower (QMIN) limits of the frequency range.
- Determine the loudness parameter (A).
- Set the reducing factors (Alpha and Landa) for loudness (A) and pulse rate (r), respectively. They are often kept constant at 1 during the algorithm.

Step 2: Lucky step - Each bat has a probability of receiving a pulse from the best bat. Generate a random number between 0 and 1 for each bat. If the generated number is greater than the pulse emission rate (ri), it means the bat successfully received the pulse from the best bat. Otherwise, it did not.

Step 3: Move and update - In this step, bats have the opportunity to move to a new position. If the new position yields a better objective function value, update the position and replace the previous solution. This step is similar to other algorithms.

The unique aspect of the bat algorithm lies in checking the loudness condition. If the following condition holds true: "If RAND < Loudness," generate a random number between 0 and 1. If the generated

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)],$$

number is smaller than the loudness value (A), the bat should move to a new position. After the movement, the loudness (A) and pulse rate (r) parameters should be reduced according to Equation (4-4).

Step 4: Update global best - After checking the position condition and updating the bat's position if necessary, update the global best solution (GPOP). The global population represents the best-fit solution. Depending on whether the problem is a minimization or maximization problem, the decision is made accordingly. For minimization, the lower merit is considered the best solution.

These three steps constitute a cycle in the main loop, and the process continues until a stopping condition is met or a predefined maximum number of iterations (usually 1000 steps) is reached.

4.8.2.5 Evaluation of the Bat Algorithm

The evaluation of algorithms generally involves two perspectives: comparing the number of function evaluations for a given accuracy and comparing the performance accuracy within a fixed number of function evaluations.

In the image below, the pattern of 25 bats is depicted over 20 iteration steps until convergence at point (1,1).

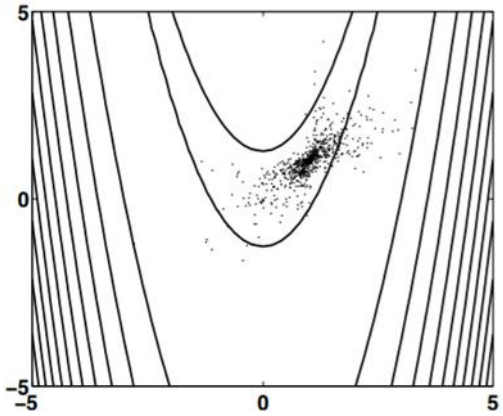


Figure 4.3. Bat Movement Pattern during 20 Iteration Steps until Convergence at Point (1,1)

The figure illustrates the iterative progress of the algorithm over 20 steps. The paths traced by the bats vividly illustrate the algorithm's gradual convergence towards the designated point (1,1). This convergence not only accentuates the effectiveness of the algorithm but also offers a visual understanding of its efficiency. Additionally, the figure effectively communicates the delicate balance between precision and computational complexity – a pivotal factor in the assessment and selection of algorithms.

By presenting the bat movement pattern, the figure visually represents the algorithm's progression over the specified iteration steps. The convergence of the bats' movements towards the point (1,1) offers insights into the effectiveness and efficiency of the algorithm being evaluated. This illustrative representation aids in comprehending the trade-off between accuracy and computational effort, which is a critical aspect of algorithm evaluation and selection.

Furthermore, the convergence function in the three-dimensional space of the placement model of 40 bats is illustrated in the last 10 stages of repetition.

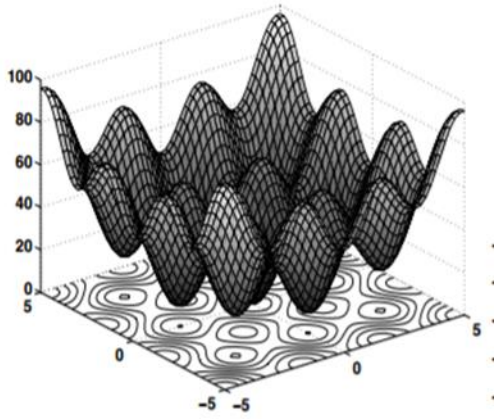


Figure 4.4. Convergence Function in Three-Dimensional Space

Within this three-dimensional expanse, the figure illustrates the comprehensive coverage achieved by 40 bats. The X-axis denotes lateral movement, traversing from left to right or right to left. The Y-axis signifies vertical motion, ascending from bottom to top or descending from top to bottom. Finally, the Z-axis captures movement perpendicular to the X-Y plane, typically extending into and out of the plane itself. This presentation underscores a 3D coordinate system, characterized by its distinct axes. Specifically, the X and Z axes are aligned at positions -5, 0, and 5, while the Y axis spans a range from 0 to 100.

The graphical representation in Figure 4.4 aids in understanding how the placement of the 40 bats evolves in a multi-dimensional space as the algorithm progresses.

The convergence of the function within this space offers valuable information about the effectiveness and reliability of the algorithm's ability to reach a stable solution. This visual insight is pivotal in assessing the algorithm's performance and its capability to optimize the placement model efficiently and accurately.

4.9 Comparison of Bat Algorithm with Other Algorithms Including GA and PSO

The Bat Algorithm (BA) has been developed by incorporating the strengths, advantages, and features of other algorithms such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), along with inspiration from the behavior of bats in nature [39], [40].

In the following table, a comparison is presented between the Bat Algorithm and GA and PSO algorithms using different benchmark functions. Based on the table, it can be observed that PSO outperforms GA in terms of performance. However, the Bat Algorithm (BA) surpasses both GA and PSO in terms of accuracy and efficiency. Thus, BA is considered a highly robust and efficient algorithm for solving optimization problems. It should be noted that the implementation of BA is more complex compared to other meta-heuristic algorithms. The results clearly demonstrate the advantages of the Bat Algorithm over its counterparts.

Table 4.2: Comparison of Three Optimization Algorithms on Ten Benchmark Functions

Functions/Algorithms	GA	PSO	BA
Multiple peaks	$52124 \pm 3277(98\%)$	$3719 \pm 205(97\%)$	$1152 \pm 245(100\%)$
Michalewicz's (d=16)	$89325 \pm 7914(95\%)$	$6922 \pm 537(98\%)$	$4752 \pm 753(100\%)$
Rosenbrock's (d=16)	$55723 \pm 8901(90\%)$	$32756 \pm 5325(98\%)$	$7923 \pm 3293(100\%)$
De Jong's (d=256)	$25412 \pm 1237(100\%)$	$17040 \pm 1123(100\%)$	$5273 \pm 490(100\%)$
Schwefel's (d=128)	$227329 \pm 7572(95\%)$	$14522 \pm 1275(97\%)$	$8929 \pm 729(99\%)$
Ackley's (d=128)	$32720 \pm 3327(90\%)$	$23407 \pm 4325(92\%)$	$6933 \pm 2317(100\%)$
Rastrigin's	$110523 \pm 5199(77\%)$	$79491 \pm 3715(90\%)$	$12573 \pm 3372(100\%)$
Easom's	$19239 \pm 3307(92\%)$	$17273 \pm 2929(90\%)$	$7532 \pm 1702(99\%)$
Griewangk's	$70925 \pm 7652(90\%)$	$55970 \pm 4223(92\%)$	$9792 \pm 4732(100\%)$
Shubert's (18 minima)	$54077 \pm 4997(89\%)$	$23992 \pm 3755(92\%)$	$11925 \pm 4049(100\%)$

4.10 Using the Bat Algorithm to Solve a Minimization Problem

As an example, let's consider the well-known "sphere=∑x^2" function, which is widely used as a test for continuous minimization problems. We will apply the Bat Algorithm with the provided data (fitness value of 66.38 and the values of x=1.2, 0.5, 6.3, -5) in MATLAB software to find the optimal solution.

The results of applying the Bat Algorithm to solve this problem are visualized in the graphs below. The red curve represents the best solution found, while the blue curve represents the average solution obtained during the optimization process. The information obtained from the four steps of executing the problem-solving code includes the best solution, the best fitness value, and the time taken to obtain the solution.

By analyzing the graphs, we can observe that the Bat Algorithm achieves higher accuracy and convergence compared to other optimization algorithms. The obtained solutions are close to each other, indicating the algorithm's effectiveness in finding optimal solutions within a shorter time frame.

Bat Algorithm

BEST solution = 9.7647e-05 -7.834e-05 -3.1628e-05 0.00013373 -5.1088e-05
BEST fitness = 3.7166e-08
Time = 1.5716

BEST solution = 6.6693e-06 -0.0002857 -3.7894e-05 0.00020181 2.3426e-0
BEST fitness = 1.2438e-07
Time = 1.2843

BEST solution = -2.2033e-05   -0.000324 -9.9306e-05 5.2769e-05 0.00014579
BEST fitness = 1.3936e-07
Time = 1.1528

BEST solution = 0.00010667 0.00024646 -0.0001039 5.9563e-05 -3.6346e-05
BEST fitness = 8.7787e-08
Time = 1.1391

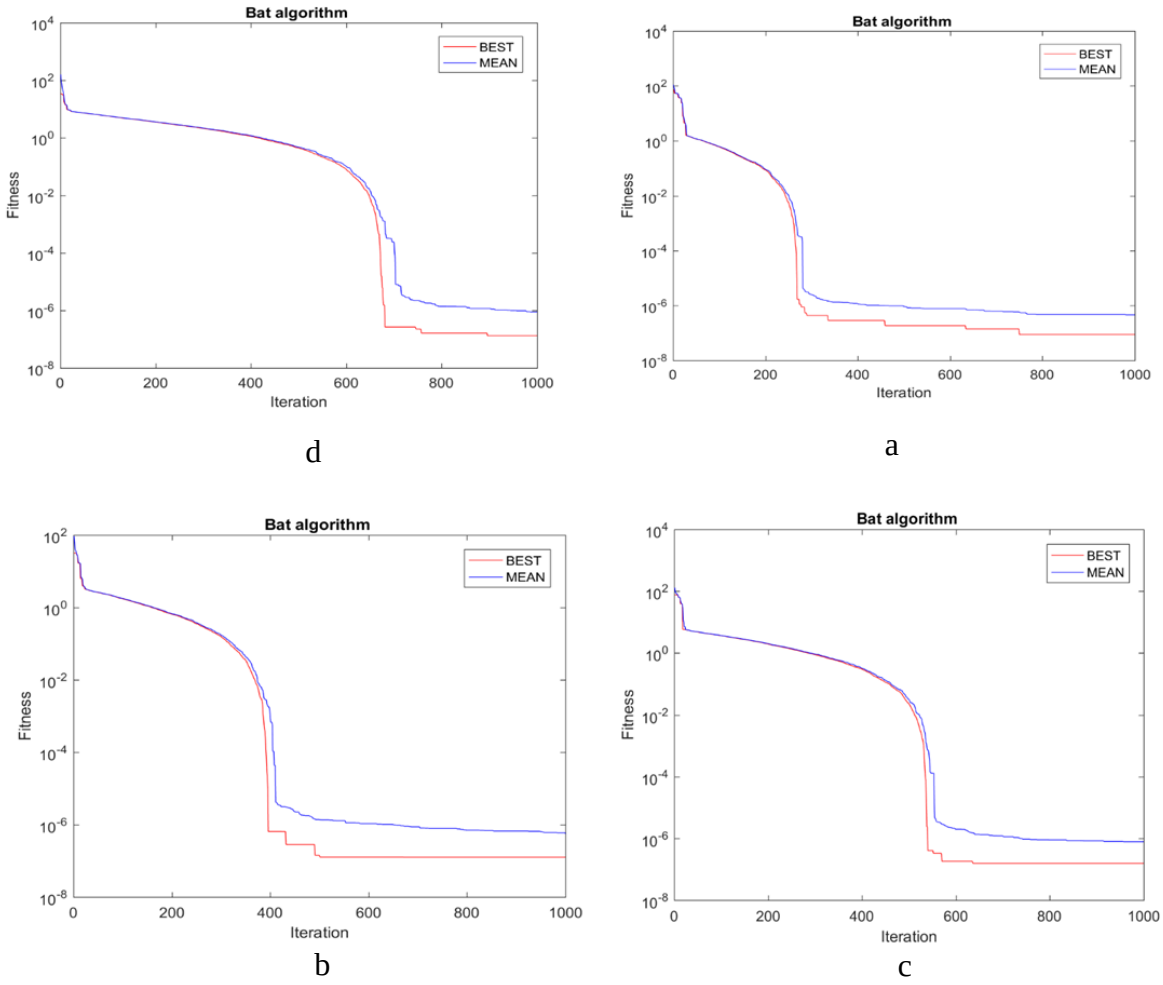


Figure 4.5. Execution of the Bat Algorithm Code for Solving the Function - 1000 Iterations

This figure demonstrates the utilization of the bat algorithm to solve a minimization problem. The red line represents the optimal solution, while the blue line depicts the algorithm's output. The x-axis denotes the 1000 iterations, and the y-axis indicates the fitness value.
In sequence a, b, c, and d, it is evident that the red and blue lines have significantly converged, and consequently, the results have closely approached the best solution.

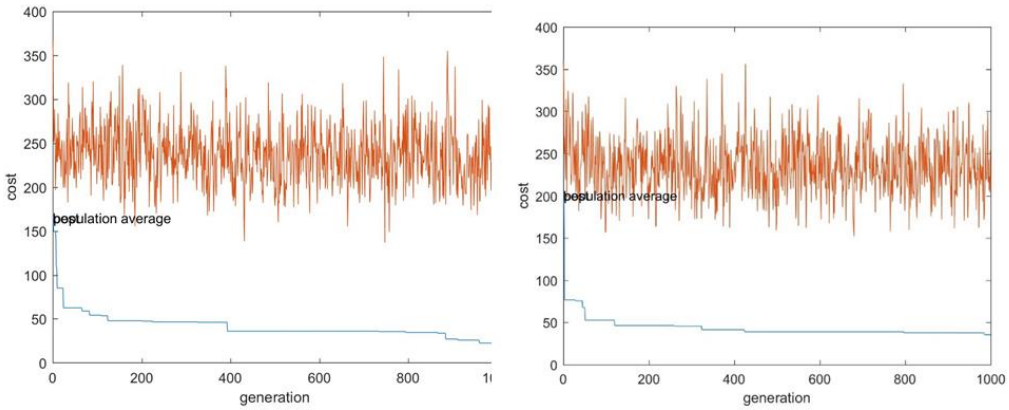
4.11 Utilizing the Genetic Algorithm for Minimization Problem Solving

The genetic algorithm was applied to solve the same minimization problem as mentioned earlier, and the following results were obtained.

continuous genetic algorithm

best solution
0.094256- 0.1733- 0.12484 0.32605 0.66063- 1.6187 0.052573- 0.40268-
RMS_FF_GA_ANN =
4.445

best solution
0.31928-0.15453 0.41428 0.10763- 1.0802- 0.21166 1.3902 0.18582-
RMS_FF_GA_ANN=
3.9046



b a

Figure 4.6. Execution of the Genetic Algorithm Code for Function Solving - 1000 Repetition Steps

In Figure 4.6, a Genetic Algorithm is executed to optimize a function over 1000 steps. The red line depicts the improvement in fitness for the best solutions over iterations, and the blue line illustrates the average fitness trends. The x-axis represents iterations, while the y-axis displays the cost.
a. The red line showcases the best population's average, and as time progresses, both the red and blue lines exhibit a descending trend in cost, stabilizing around 50.
b. The cost rapidly decreases to below 50 and further approaches zero in a shorter time frame.

4.12 Sending GPS-Based Data from a Robot Using Genetic Algorithm and Bat

Mobile robots are extensively employed in various domains, including communication and transportation, necessitating the use of motion sensors such as GPS, accelerometers, gyroscopes, and more. To ensure the efficiency of the entire network, it becomes crucial to employ an optimization algorithm capable of accurately and efficiently transmitting the available dataset without any alterations or modifications.

In this research, the existing GPS output dataset has been integrated into the code of the genetic algorithm and bat algorithm within the MATLAB software. Based on previous studies discussed in the preceding chapters, these two algorithms have been chosen for this research. The following section presents the results obtained using the genetic optimization algorithm.

continuous genetic algorithm

Best Solution
0.30815 0.18886 0.92624- 0.080345- 0.80882- 0.38833 0.094368 1.0478
RMS_FF_GA_ANN= 44.778

Best Solution
0.3465- 0.018264- 0.0018161 1.3334 1.101- 0.80785 0.027942- 0.57448-
RMS_FF_GA_ANN= 42.389

Best Solution
- 0.071941- 0.77002 0.30856- 0.40065- 1.7413 0.12279- 0.0075249- 0.37833
RMS_FF_GA_ANN= 41.042

Best Solution
0.16244- 0.47362 0.14859- 0.91953- 1.1291- 0.0067779 0.26593 1.9916
RMS_FF_GA_ANN= 29.017

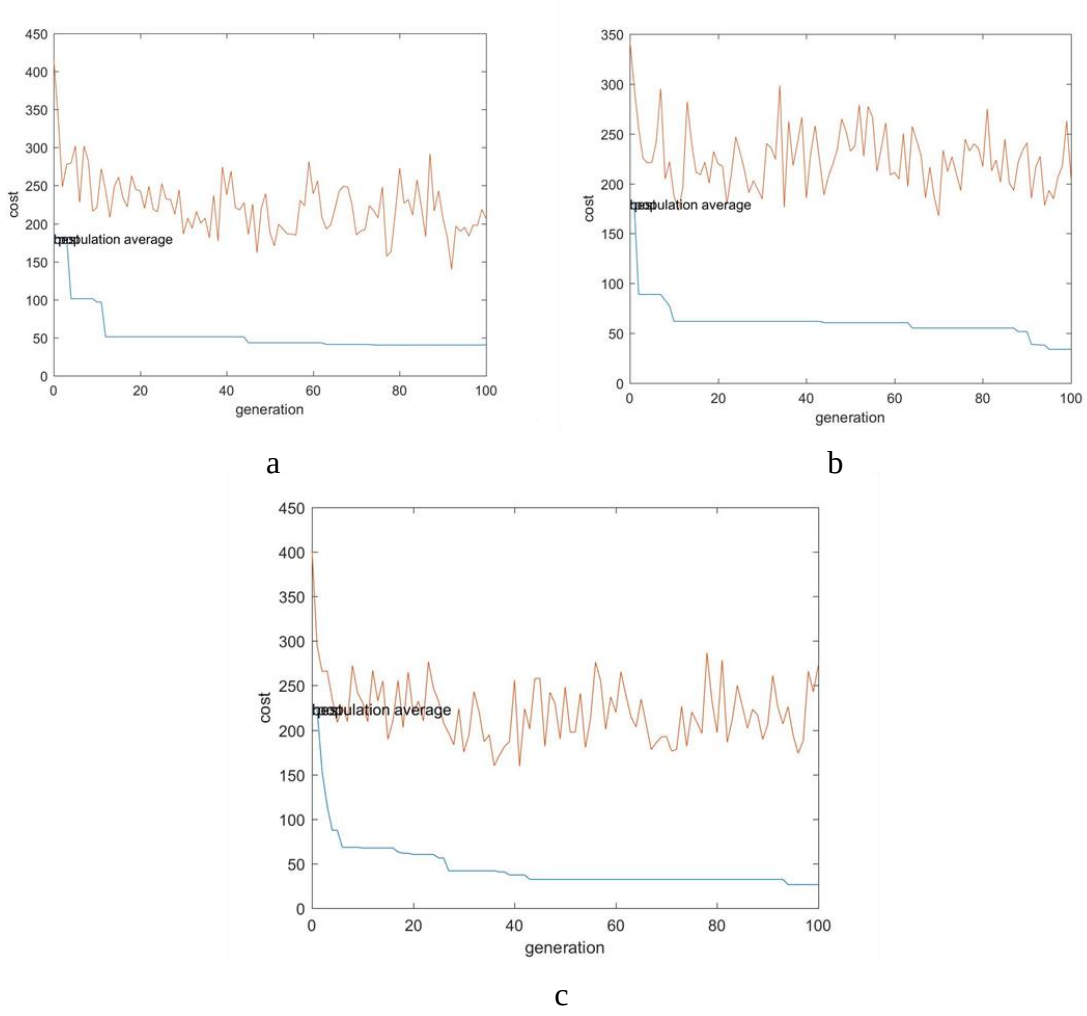


Figure 4.7. Data Transmission Optimization using Genetic Algorithm with GPS in 100 Repetition Steps

These figures display the utilization of GPS data for this research and the application of a genetic algorithm across 1000 iterations. The red line represents the optimal solutions, while the blue line illustrates the results at each iteration. The x-axis corresponds to the iterations, and the y-axis represents the cost.
a. The red line presents the average of the best population's performance. As time advances, both the red and blue lines demonstrate a declining trend in cost, eventually stabilizing around 50.
b. The cost rapidly decreases to below 50, coinciding with a reduction in the best solution. The convergence of the red and blue lines reflects this decreasing trend.
c. The cost rapidly drops to below 30, accompanied by a continued decrease in the best solution. The proximity of the red and blue lines underscores this convergence.

The results obtained using the bat algorithm are as follows:

BEST solution = 1.0935 0.17215 -2.405 -1.0213 1.9514 -4.1492 -0.19867 2.2593
BEST fitness = 226.6155
Time = 109.0036

BEST solution = -1.9563 -0.78336 -5.6846 0.54317 2.2657 -4.7852 , 6.3008 2.2799
BEST fitness = 271.4476
Time = 107.9067

BEST solution = 3.0765 0.56444 -4.4944 -2.9893 1.5136 -1.9662 ,1.8802 0.45476
BEST fitness = 135.9708
Time = 117.8693

BEST solution = -3.4982 -0.1027 5.9437 1.9568 -1.8585 1.6108 ,1.1419 -0.85598
BEST fitness = 204.4738
Time = 133.2839

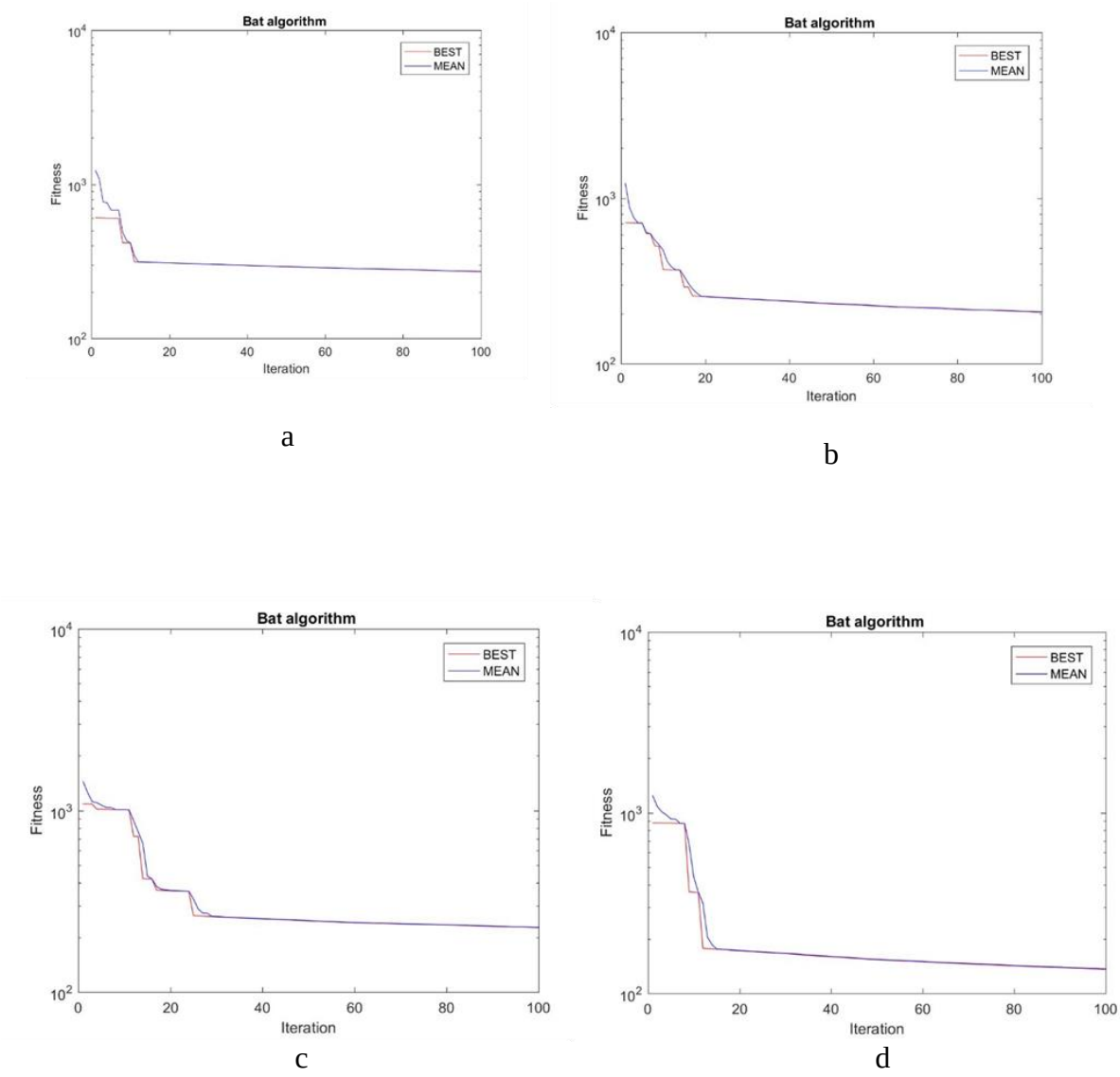


Figure 4.8. Data Transmission Optimization using Bat Algorithm with GPS in 100 Repetition Steps

These figures depict the utilization of GPS data for the bat algorithm. The blue line represents the average results, while the red line illustrates the best solution. Iterations are displayed on the X-axis, and the fitness values are plotted on the Y-axis.
From point "a" to point "d," the fitness value decreases due to the nature of solving a minimization problem. Additionally, both the blue and red lines converge, drawing closer together.

4.13 Conclusion and Comparison of Used Algorithms

Both algorithms were evaluated in terms of the number of iteration steps and the same fitness using a similar dataset. The results obtained from executing the optimization algorithm commands in MATLAB software indicate that there are clear differences between the two algorithms.

The genetic algorithm achieved the desired response range in a shorter time, while the bat optimization algorithm took more time in the same 100 iteration steps. However, the bat algorithm produced results with data points that were closer to each other, indicating higher accuracy in the provided answers. It achieved a higher approximation to the optimal response range compared to the genetic algorithm, which can be considered as a positive aspect.

Considering the specific conditions and limitations of the problem implementation, it is important to choose the appropriate algorithm for optimization. If time efficiency is crucial, especially in scenarios involving routing, data transmission, and positioning, the genetic algorithm can be preferred. On the other hand, if high accuracy is required for pathfinding, network leader selection, and precise relative and absolute positioning of robots, the bat algorithm is recommended.

Overall, the choice of algorithm should be based on the specific needs and requirements of the network of robots, considering factors such as time constraints and the desired level of accuracy.

5 Conclusion

In conclusion, this thesis focused on optimization problems and the application of metaheuristic algorithms, specifically the Bat Algorithm and Genetic Algorithm, in solving these problems. The thesis explored the use of rearrangement operators, such as the pseudo-dominant truncation operator, to rearrange sequences and find optimal solutions.

The selection and comparison of optimization algorithms were discussed, emphasizing the need for careful consideration of algorithm limitations and conditions when choosing the most suitable approach for a specific problem. The use of metaheuristic algorithms inspired by nature, such as genetics or ant colony optimization, was highlighted for their ability to resemble real-world scenarios and emulate natural phenomena.

The thesis examined two specific metaheuristic algorithms, the Bat Algorithm and Genetic Algorithm, in detail. These algorithms were chosen based on their resemblance to movement mechanisms and data exchange processes observed in ground and aerial robotic systems. The Genetic Algorithm simulated gradual evolution through patterns like selection, crossover, mutation, and replacement, while the Bat Algorithm drew inspiration from the flight behavior and reflexive behaviors of bats.

The Bat Algorithm, in particular, demonstrated its effectiveness in optimization tasks by utilizing characteristics like position, velocity, and objective function value to guide the movement of bats towards desired destinations. The algorithm's steps, including lucky step, move and update, and update global best, were outlined to showcase its iterative nature.

Comparisons were made between the Bat Algorithm, Genetic Algorithm, and Particle Swarm Optimization (PSO) using benchmark functions, revealing the superior accuracy and efficiency of the Bat Algorithm. While PSO outperformed the Genetic Algorithm, the Bat Algorithm showcased its robustness and effectiveness in solving optimization problems.

Lastly, a practical example involving the minimization of the "sphere=∑x^2" function was presented, where the Bat Algorithm was applied to find the optimal solution. The results demonstrated the algorithm's ability to converge quickly and achieve high accuracy compared to other optimization algorithms.

Overall, this thesis contributes to the field of optimization by providing insights into the application of metaheuristic algorithms, specifically the Bat Algorithm and Genetic Algorithm. These algorithms offer effective solutions to optimization problems, showcasing their potential in various real-world scenarios. Further research and exploration of these algorithms can lead to advancements in optimization techniques and improved problem-solving capabilities.

6 References

[1] Yao, Y., He, N., & Zhang, M. (2022). The Application of Internet of Things in Robot Route Planning Based on Multisource Information Fusion. Computational Intelligence and Neuroscience, 2022.

[2] Matsuda, Y., Sato, Y., Sugi, T., Goto, S., & Egashira, N. (2022). Control system for object transportation by a mobile robot with manipulator combined with manual operation and autonomous control. Int J Innov Comput Inf Control, 18(2), 621-631.

[3] Kliestik, T., Musa, H., Machova, V., & Rice, L. (2022). Remote Sensing Data Fusion Techniques, Autonomous Vehicle Driving Perception Algorithms, and Mobility Simulation Tools in Smart Transportation Systems. Contemporary Readings in Law and Social Justice, 14(1), 137-152.

[4] Jiang, M., Chen, Y., Zheng, W., Wu, H., & Cheng, L. (2016, August). Mobile robot path planning based on dynamic movement primitives. In 2016 IEEE International Conference on Information and Automation (ICIA) (pp. 980-985). IEEE.

[5] Hicham , T., Hicham, H.-A., Nicolas , S., & Bouchafa, S. (2021). Multi-Robot Task Sequencing & Automatic Path Planning for Cycle Time Optimization: Application for Car Production Line. IEEE ROBOTICS AND AUTOMATION LETTERS,, 6(2), 1335 - 1342. doi:10.1109/LRA.2021.3057011

[6] Chen, W., Liu, Z., Zhao, H., Zhou, S., Li, H., & Liu, Y.-H. (2020). CUHK-AHU Dataset: Promoting Practical Self-Driving Applications in the Complex Airport Logistics, Hill and Urban Environments. 2020 IEEE/RSJ International Conference

on Intelligent Robots and Systems (IROS). Las Vegas, NV, USA. doi:10.1109/IROS45743.2020.9341317.

[7] Sánchez-Ibáñez, J., Carlos J. , P.-d.-P., & García-Cerezo, A. (2021). Path Planning for Autonomous Mobile Robots: A Review. Space Robotics Laboratory, Department of Systems Engineering and Automation, Universidad de Málaga. doi:10.3390/s21237898.

[8] Hazha , S., & Salih Mohammed , A. (2022). Path planning optimization in unmanned aerial vehicles using meta-heuristic algorithms: a systematic review. Environmental Monitoring and Assessment.

[9] Poliak, M., Jurecki, R., & Buckner, K. (2022). Learning Object Detection Technology in Smart Sustainable Urban Transport Systems. Contemporary Readings in Law and Social Justice, 25-40.

[10]Lu , X., Li, H., Zhang , X., Gao, B., & Cheng, T. (n.d.). Magnetic-assisted self-powered acceleration sensor for real-time monitoring vehicle operation and collision based on triboelectric nanogenerator.

[11] Abid, M., Tabaa, M., Chakir, A., & Hachimi, H. (2022). Routing and charging of electric vehicles: Literature review. Energy Reports, 556-578. Retrieved from https://doi.org/10.1016/j.egyr.2022.07.089.

[12] Punyavathi , G., Neeladri , M., & Mahesh, K. (2022). Vehicle tracking and detection techniques using IoT. About the journal, 51(1), 909-913. Retrieved from https://doi.org/10.1016/j.matpr.2021.06.283

[13] Misra & Per , 2006Shi, C., Luo, X., Qi, P., Li, T., Song, S., Najdovski, Z., . . . Ren, H. (2017). Shape Sensing Techniques for Continuum Robots in Minimally Invasive Surgery: A Survey. IEEE Transactions on Biomedical Engineering, 64(8), 1665 - 1678. doi:10.1109/TBME.2016.2622361.

[14] Gupta, S., Morris , D., Patel, S., & Tan, D. (2012). Soundwave: using the doppler effect to sense gestures. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, .

[15] Craig, G. (2007). Rigorous 3D error analysis of kinematic scanning LIDAR systems. " Journal of Applied Geodesy jag, 1(3), 147-157.

[16] Olivier, M. (2004). Cyberbotics Ltd. Webots™: professional mobile robot simulation. International Journal of Advanced Robotic Systems, 5.

[17] H. Nair, S., Tseng, E. H., & Borrel, F. (2022). Collision Avoidance for Dynamic Obstacles with Uncertain Predictions using Model Predictive Control.

[18] Kakosimos, P., & Abu-Rub, H. (2017). Predictive Speed Control With Short Prediction Horizon for Permanent Magnet Synchronous Motor Drives. IEEE Transactions on Power Electronics, 31(3), 2740 - 2750. doi: 10.1109/TPEL.2017.2697971

[19] Misra & Per , 2006Shi, C., Luo, X., Qi, P., Li, T., Song, S., Najdovski, Z., . . . Ren, H. (2017). Shape Sensing Techniques for Continuum Robots in Minimally Invasive Surgery: A Survey. IEEE Transactions on Biomedical Engineering, 64(8), 1665 - 1678. doi:10.1109/TBME.2016.2622361.

[20] McGuire, K., Guido de , C., & Tuyls , K. (2019). A comparative study of bug algorithms for robot navigation. Robotics and Autonomous Systems,, 121. Retrieved from https://doi.org/10.1016/j.robot.2019.103261

[21] Zhong, J., Li, B., Li, S., Yang, F., Li, P., & Cui, Y. (2021). Particle swarm optimization with orientation angle-based grouping for practical unmanned surface vehicle path planning. Applied Ocean Research, 111. Retrieved from https://doi.org/10.1016/j.apor.2021.102658

[22] Shuzhi Sam, G., & Juan Cui, Y. (2000). New potential functions for mobile robot path planning. IEEE Transactions on robotics and automation, 16(5), 615-620.

[23] Misra & Per , 2006Shi, C., Luo, X., Qi, P., Li, T., Song, S., Najdovski, Z., . . . Ren, H. (2017). Shape Sensing Techniques for Continuum Robots in Minimally Invasive Surgery: A Survey. IEEE Transactions on Biomedical Engineering, 64(8), 1665 - 1678. doi:10.1109/TBME.2016.2622361

[24] Olivier, M. (2004). Cyberbotics Ltd. Webots™: professional mobile robot simulation. International Journal of Advanced Robotic Systems, 5.

[25] Warren, C. W. (1989). Global path planning using artificial potential fields. IEEE International Conference Robotics and Automation,.

[26] Bekmezci, I., Koray Sahingoz, O., & Şamil , T. (2013). Flying ad-hoc networks (FANETs): A survey." Ad Hoc Networks. 11(3), 1254-1270.

[27] Perkins, C. (2001). Ad hoc networking. Addison-wesley,.

[28] Misra, P., & Per , E. (2006). Global Positioning System: signals, measurements and performance . Massachusetts: Ganga-Jamuna Press.

[29] Xin-She , Y. (2010). Nature-Inspired Metaheuristic Algorithms. Luniver Press.

[30] Tu, J., & Simon , Y. (2003). Genetic algorithm based path planning for a mobile robot. ICRA'03. IEEE International Conference Robotics and Automation,.

[31] Burke, J., , Lewis, A. S., & Overton, M. L. (2005). A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. SIAM Journal on Optimization, 15(3), 751-779. Retrieved from https://doi.org/10.1137/0306012.

[32] Bortoff, S. (2000). Path planning for UAVs. American Control Conference, 1.

[33] Jia, D., & Juris , V. (2004). Parallel evolutionary algorithms for UAV path planning. AIAA 1st Intelligent Systems Technical Conference.

[34] Warren, C. W. (1989). Global path planning using artificial potential fields. IEEE International Conference Robotics and Automation,.

[35] Bekmezci, I., Koray Sahingoz, O., & Şamil , T. (2013). Flying ad-hoc networks (FANETs): A survey." Ad Hoc Networks. 11(3), 1254-1270.

[36] Utkarsh , A., Jatin , A., Rahul , S., Deepak , G., Ashish , K., & Aditya , K. (2019). Hybrid Wolf-Bat Algorithm for Optimization of Connection Weights in Multi-layer Perceptron. ACM Transactions on Multimedia Computing, Communications, and Applications, 16(15), 1-20. Retrieved from https://doi.org/10.1145/3350532.

[37] Wang, G., & Lihong , G. (2013). A novel hybrid bat algorithm with harmony search for global numerical optimization. Journal of Applied Mathematics.

[38] Tu, J., & Simon , Y. (2003). Genetic algorithm based path planning for a mobile robot. ICRA'03. IEEE International Conference Robotics and Automation,.

[39] Han, W., Seung-Min, B., & Tae-Yong, K. (1997). "Genetic algorithm based path planning and dynamic obstacle avoidance of mobile robots. Computational Cybernetics and Simulation. IEEE International Conference on. Vol. 3.

[40] Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. Nature inspired cooperative strategies for optimization (NICSO 2010)., 65-74.

APPENDIX A

MATLAB Code- Bat Algorithm

The provided code block is an implementation of the Bat Algorithm for optimization problems.

```
clc
clear all
close all
format shortG

%% Insert Data

% data=InsertData();

nvar=8;       % Number of Variables

lb=-10*ones(1,nvar);    % Lower Bound
ub= 10*ones(1,nvar);    % Upper Bound

%% Parameters Setting

npop=20;        % Population size
maxiter=100;   % Number of generations
A=0.9;          % Loudness  (constant or decreasing)
r=0.5;         % Pulse rate (constant or decreasing)
Qmin=0;        % Frequency minimum
Qmax=0.5;      % Frequency maximum

Alpha=0.99;    % Reduction factor A
Lambda=0.01;   % Reduction factor r


data.lb=lb;
data.ub=ub;
data.nvar=nvar;
```

```
%% Initialize the population/solutions

tic
emp.v=[];
emp.x=[];
emp.fit=[];

pop=repmat(emp,npop,1);


for i=1:npop
    pop(i).x = unifrnd(lb,ub);
    pop(i).v = 0;
    pop(i) = fitness(pop(i),data);
end


% Find the initial gpop solution
[AA,ind]=min([pop.fit]);
gpop=pop(ind);


%% Main Loop
BEST=zeros(maxiter,1);
MEAN=zeros(maxiter,1);

A=A*ones(npop,1);
r=r*ones(npop,1);
r0=r;

for iter=1:maxiter

    % Loop over all bats/solutions

    for i=1:npop

        Q=unifrnd(Qmin,Qmax); % Random Frequency

        newpop.v=pop(i).v+(pop(i).x-gpop.x)*Q; % Update Velocity

        newpop.x=pop(i).x+newpop.v;          % Update Position
```

```
        % Pulse rate
        if rand>r(i)
           newpop.x=gpop.x+0.001*randn(1,nvar);  % The factor 0.001 limits the step sizes
of random walks
        end


        newpop= fitness(newpop,data);


        % Update if the solution improves, or not too loud
        if (newpop.fit<=pop(i).fit) && (rand<A(i))
           pop(i)=newpop;
           r(i)=r0(i)*(1-exp(-Lambda*iter));
           A(i)=A(i)*Alpha;
        end

        % Update the gpop solution
        if (newpop.fit<=gpop.fit)
           gpop=newpop;
        end

    end


    BEST(iter)=gpop.fit;
    MEAN(iter)=mean([pop.fit]);

    disp(['Iter ' num2str(iter) ' BEST = ' num2str(BEST(iter))]);


end

%% Results

disp(' ')
disp([ ' BEST solution = '  num2str(gpop.x)]);
disp([ ' BEST fitness = '  num2str(gpop.fit)]);
disp([ ' Time = '  num2str(toc)]);

figure
semilogy(BEST,'r');
hold on
semilogy(MEAN,'b');
```

```
xlabel('Iteration');
ylabel('Fitness');
legend('BEST','MEAN');
title('Bat algorithm ')
```

CB

```
function x=CB(x,lb,ub)

    x=max(x,lb);
    x=min(x,ub);

end
```

output

```
BEST solution = 9.7647e-05  -7.834e-05 -3.1628e-05  0.00013373 -5.1088e-05
BEST fitness = 3.7166e-08
Time = 1.5716

BEST solution = 6.6693e-06  -0.0002857 -3.7894e-05  0.00020181  2.3426e-0
BEST fitness = 1.2438e-07
Time = 1.2843

BEST solution = -2.2033e-05   -0.000324 -9.9306e-05  5.2769e-05  0.00014579
BEST fitness = 1.3936e-07
Time = 1.1528

BEST solution = 0.00010667  0.00024646  -0.0001039  5.9563e-05 -3.6346e-05
BEST fitness = 8.7787e-08
Time = 1.1391
```

Fitness

```
function sol=fitness(sol,data)

lb=data.lb;
ub=data.ub;


x=sol.x;
x=CB(x,lb,ub);

% sol.fit=((x-1).^2)+1;
```

```
sol.x=x;

load GA_inputs;
parNum=2;
target=target(parNum,:);
outputTest=outputTest(parNum,:);

net=newff(inputTrain,target);

for i=1%:size(x,1)%%%%npar=7; % number of optimization variables
   net.iw{1,:}=x(i,1:7);
   net.b{1,1}=x(i,8);
   net_output= sim(net,inputTrain);
   sol.fit(i,1)=mean(sqrt(mean(((net_output-target).^2),2)));
end

end
end
```

APPENDIX B

MATLAB Code- Genetic Algorithm

The provided code block is an implementation of the Genetic Algorithm for optimization problems.

```
% Continuous Genetic Algorithm
%
% minimizes the objective function designated in ff
% Before beginning, set all the parameters in parts
% I, II, and III
% 2023
%_____________________________________________________
% I Setup the GA
clc
clear all
parNum=2;

load GA_inputs;
inputTrain=inputTrain(parNum,:);
inputTest=inputTest(parNum,:);
target=target(parNum,:);
outputTest=outputTest(parNum,:);

ff='optimization_func_onePar'; % objective function
npar=8; % number of optimization variables
varhi=2; varlo=-2; % variable limits
%_____________________________________________________
% II Stopping criteria
maxit=100; % max number of iterations
mincost=0; % minimum cost
%_____________________________________________________
% III GA parameters
popsize=10; % set population size
mutrate=.5; % set mutation rate
selection=0.5; % fraction of population kept
Nt=npar; % continuous parameter GA Nt=#variables
keep=floor(selection*popsize); % #population
% members that survive
nmut=ceil((popsize-1)*Nt*mutrate); % total number of
% mutations
```

```
M=ceil((popsize-keep)/2); % number of matings
%_____________________________________________________
% Create the initial population
iga=0; % generation counter initialized
par=(varhi-varlo)*rand(popsize,npar)+varlo; % random
cost=feval(ff,par); % calculates population cost
% using ff
[cost,ind]=sort(cost); % min cost in element 1
par=par(ind,:); % sort continuous
minc(1)=min(cost); % minc contains min of
meanc(1)=mean(cost); % meanc contains mean of population
%_____________________________________________________
% Iterate through generations
while iga<maxit
   iga=iga+1; % increments generation counter
   %_____________________________________________________
   % Pair and mate
   M=ceil((popsize-keep)/2); % number of matings
   prob=flipud([1:keep]'/sum([1:keep])); % weights
   % chromosomes
   odds=[0 cumsum(prob(1:keep))']; % probability
   % distribution
   % function
   pick1=rand(1,M); % mate #1
   pick2=rand(1,M); % mate #2
   % ma and pa contain the indicies of the chromosomes
   % that will mate
   ic=1;
   while ic<=M
      for id=2:keep+1
         if pick1(ic)<=odds(id) & pick1(ic)>odds(id-1)
            ma(ic)=id-1;
         end
         if pick2(ic)<=odds(id) & pick2(ic)>odds(id-1)
            pa(ic)=id-1;
         end
      end
      ic=ic+1;
   end
   %_____________________________________________________
   % Performs mating using single point crossover
   ix=1:2:keep; % index of mate #1
   xp=ceil(rand(1,M)*Nt); % crossover point
   r=rand(1,M); % mixing parameter
   for ic=1:M
      xy=par(ma(ic),xp(ic))-par(pa(ic),xp(ic)); % ma and pa
      % mate
      par(keep+ix(ic),:)=par(ma(ic),:); % 1st offspring
```

```
      par(keep+ix(ic)+1,:)=par(pa(ic),:); % 2nd offspring
      par(keep+ix(ic),xp(ic))=par(ma(ic),xp(ic))-r(ic).*xy;
      % 1st
      par(keep+ix(ic)+1,xp(ic))=par(pa(ic),xp(ic))+r(ic).*xy;
      % 2nd
      if xp(ic)<npar % crossover when last variable not selected
         par(keep+ix(ic),:)=[par(keep+ix(ic),1:xp(ic))...
            par(keep+ix(ic)+1,xp(ic)+1:npar)];
         par(keep+ix(ic)+1,:)=[par(keep+ix(ic)+1,1:xp(ic))...
            par(keep+ix(ic),xp(ic)+1:npar)];
      end % if
   end
   %_____________________________________________________________
   % Mutate the population
   mrow=sort(ceil(rand(1,nmut)*(popsize-1))+1);
   mcol=ceil(rand(1,nmut)*Nt);
   for ii=1:nmut
      par(mrow(ii),mcol(ii))=(varhi-varlo)*rand+varlo;
      % mutation
   end % ii
   %_____________________________________________________________
   % The new offspring and mutated chromosomes are
   % evaluated
   cost=feval(ff,par);
   %_____________________________________________________________
   % Sort the costs and associated parameters
   [cost,ind]=sort(cost);
   par=par(ind,:);
   %_____________________________________________________________
   % Do statistics for a single nonaveraging run
   minc(iga+1)=min(cost);
   meanc(iga+1)=mean(cost);
   %_____________________________________________________________
   % Stopping criteria
   if iga>maxit | cost(1)<mincost
      break
   end
   [iga cost(1)]
end %iga

%_____________________________________________________________
% Displays the output
day=clock;
disp(datestr(datenum(day(1),day(2),day(3),day(4),day(5),day(6)),0))
disp(['optimized function is ' ff])
format short g
disp(['popsize = ' num2str(popsize) ' mutrate =' num2str(mutrate) '# par =' num2str(npar)])
disp(['#generations=' num2str(iga) ' best cost=' num2str(cost(1))])
```

```
disp(['best solution'])
disp([num2str(par(1,:))])
disp('continuous genetic algorithm')
figure(24)
iters=0:length(minc)-1;
plot(iters,minc,iters,meanc);
xlabel('generation');ylabel('cost');
text(0,minc(1),'best');text(1,minc(2),'population average')

%%
% % % % % % % % FF

load GA_inputs;
target=target(parNum,:);
outputTest=outputTest(parNum,:);
%
net=newff(inputTrain,target);
x=par(1,:);
i=1;
net.iw{1,:}=x(i,1:7);
net.b{1,1}=x(i,8);
% net_output(i,:)= sim(net,inputTrain);
FF_GA_ANN= sim(net,inputTest);
RMS_FF_GA_ANN=sqrt(mean(((FF_GA_ANN-outputTest).^2),2))

R2_FF_GA_ANN=1-(sum(FF_GA_ANN-outputTest).^2)/(sum(outputTest)^2);
MAPE__FF_GA=(1/size(outputTest,2))*(sum(FF_GA_ANN-
outputTest))/(sum(outputTest));


optimization_func_onePar(x)

function [cost] = optimization_func_onePar(x)

load GA_inputs;
parNum=2;
target=target(parNum,:);
outputTest=outputTest(parNum,:);

net=newff(inputTrain,target);

cost=zeros(10,1);
for i=1:size(x,1)%%%%npar=7; % number of optimization variables
   net.iw{1,:}=x(i,1:7);
   net.b{1,1}=x(i,8);
   net_output= sim(net,inputTrain);
   cost(i,1)=mean(sqrt(mean(((net_output-target).^2),2)));
end
```

```
output
best solution
-0.40268   -0.052573     1.6187    -0.66063     0.32605     0.12484    -0.1733   -0.094256
RMS_FF_GA_ANN =
     4.445

best solution
0.31928 -   -0.18582     1.3902     0.21166    -1.0802    -0.10763     0.41428     0.15453
RMS_FF_GA_ANN =
     3.9046
```

APPENDIX C

Date Set

input test (7*9 double)								
0.333333	0	0.333333	0.333333	1	1	1	0.667	0.333333
0.5	0	1	0.5	0	1	0.5	0	1
0.192811	0.223363	0.352246	0.490629	0.823363	1	0.918357	0.74095	0.362003
0.436244	0.04158	0.366944	0.105683	0.446639	1	0.61781	0.12578	0.372141
0.330522	0.000803	0.330522	0.330522	0.999197	1	1	0.663454	0.330522
0.869085	0.919558	0.746057	0.706625	0.629338	0.807571	0.771293	0.648265	0.886435
0.256829	0.291351	0.431715	0.56525	0.853187	1	0.927921	0.785281	0.445372

Input train (7*36 double)

0.333333	0.5	0.305777	0.268884	0.330522																																
0.333333	0	0.259564	0.171171	0.330522																																
0.666333	1	0.758922	0.190575	0.663454																																
0.667	0.5	0.753787	0.169785	0.662249																																
0.666333	1	0.768935	0.237006	0.661847																																
0.667	0.5	0.7638	0.203742	0.663454																																
0.667	0	0.75353	0.151074	0.663454																																
1	1	0.987677	0.740125	1																																
1	0	0.861874	0.502772	1																																
1	1	0.858793	0.883576	1																																
1	0.5	0.730167	0.689189	1																																
1	0	0.640308	0.543659	1																																
1	0.5	0.897047	0.876992	1																																
1	0	0.727599	0.672557	1																																
1	1	0.902696	0.616424	1																																
1	0.5	0.856483	0.525988	0.998795																																
0.333333	1	0.737612	0.045392	0.330522																																
0.333333	0.5	0.735302	0.020097	0.330522																																
0.333333	0	0.735302	-1.39E-17	0.330522																																
0.333333	1	0.495764	0.135482	0.330522																																
0.333333	0	0.485751	0.078309	0.330522																																
0.333333	1	0.491399	0.125087	0.330522																																
0.333333	0.5	0.491656	0.100832	0.330522																																
0.333333	0	0.486264	0.067568	0.330522																																
0.333333	0.5	0.303466	0.278933	0.330522																																
0.333333	0	0.267522	0.215523	0.330522																																
0	1	0.238511	0.093209	0.000402																																
0	0.5	0.233376	0.076923	0.000803																																
0	1	0.222593	0.053015	0.000402																																
0	0.5	0.22285	0.031532	0.000803																																
0	0	0.223107	0.028413	0.000402																																
0.333333	1	0.267266	0.504158	0.330522																																
0.333333	0	0.131194	0.362786	0.330522																																
0	1	0.076765	0.278933	0.001606																																
0	0.5	0.045956	0.22869	-1.78E-15																																
0	0	0	0.134442	-1.78E-15																																

0.723975	0.958991	0.843849	0.758675	0.917981	0	0.723975	0.515773	0.739748
0	0.067147	0.108877	0.182473	0.344841	0.280349	0.278452	0.281487	0.298558

0.858044	0.649842	0.731861	0.76183	0.692429	0.615142	0.812303	0.649842	0.641956
0.303869	0.335357	0.378604	0.554628	0.559181	0.561457	0.560698	0.572079	0.774279

0.564669	0.785489	0.750789	0.711356	0.709779	0.809148	0.695584	0.676656	0.763407
0.775038	0.781108	0.878604	0.91654	0.778073	0.924886	0.703338	0.781866	0.89302

0.736593	0.626183	0.891167	0.903785	0.753943	0.649842	0.769716	1	0.981073
0.881639	0.981032	0.792489	0.800835	0.806146	0.795144	0.797041	0.329666	0.382398

output (5*9 double)

4.429	1.022	3.92	1.907	5.764	10.199	6.671	2.187	4.69
281	376.9	378.1	479	579.6	482.4	578.6	578.5	378.9
277.83	286.46	285.5	292.32	297.85	293.17	298.75	297.89	286.77
284.5	283.79	288.25	291.98	301.57	302.61	303.34	299.03	287.94
283.54	282.78	287.24	290.83	300.44	301.52	302.35	298.03	287.05

Target (5*36 double)	3.594	379	286.99	287.39	286.54
	2.571	379	287.13	287.38	286.66
	2.96	580.5	298.8	300.57	299.19
	2.698	580.5	298.05	299.69	298.58
	2.976	579.4	298.62	300.43	299.03
	2.648	579.4	299.6	299.72	298.56
	2.139	579.4	299.5	299.24	298.25
	7.816	579.6	297.84	304.24	303.19
	5.554	579.6	298.54	301.98	300.95
	10.562	479.4	292.67	301.99	300.89
	8.362	480.3	292.21	300.54	299.56
	6.749	479.3	292.27	299	298
	9.061	479.3	293	300.96	299.96
	6.955	478.3	292.3	298.56	297.67
	7.717	579.5	298.35	304.22	303.16
	6.717	579.5	298.6	303.12	302.09
	1.087	579.2	298.84	295.55	293.09
	0.84	578.3	297.39	295.73	293.15
	0.655	579.3	297.92	295.65	293.18
	2.237	478	291.9	292.31	291
	1.597	479.1	292.99	291.08	290.17
	1.787	477.3	291.65	292.97	291.54
	1.582	479.4	292.22	292.41	291.21
	1.291	479.3	292.68	291.92	290.94
	3.163	379.1	285.47	287.77	286.89
	2.547	377.1	284.75	287.02	286.25
	1.445	378.8	286.18	284.73	283.45
	1.29	378.8	285.44	284.07	283.06
	1.273	376.6	283.95	284.82	283.56
	1.072	377.7	285.26	284.53	283.46
	1.006	377.8	280.69	283.45	282.79
	5.12	292	279.19	286.17	285.14
	3.749	278	276.84	282.74	281.81
	3.048	278.8	277.41	279.63	278.91
	2.61	278.8	278.17	278.75	278.03
	1.772	278.9	276.46	279.37	278.72