A Cloud-Assisted Reliable Trust Computing Scheme for Data Collection in Internet of Things

A PROJECT REPORT

Submitted by

AKASH SK -422420104301

MADHANKUMAR R -422420104018

REGUNATHAN V -422420104029

SRITHARAN T -422420104036

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING TINDIVANAM

ANNA UNIVERSITY: CHENNAI 600025

MAY 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"A CLOUD-ASSISTED RELIABLE TRUST COMPUTING SCHEME FOR DATA COLLECTION IN INTERNET OF THINGS"** is the Bonafide work of **"AKASH SK (422420104301), MADHANKUMAR R (422420104018), REGUNATHAN V (422420104029), SRITHARAN T (422420104036)"** who carried out the project work under my supervision.

SIGNATURE

Dr. L. Jegatha Deborah, M.E., Ph. D,

HEAD OF THE DEPARTMENT

Assistant professor,

Department of CSE,

University College of Engineering,

Tindivanam,

Melpakkam – 604001

SIGNATURE

MS. P. Sathya, M.Tech,

SUPERVISOR

Teaching Fellow,

Department of CSE,

University College of Engineering,

Tindivanam,

Melpakkam – 604001

Submitted for the University Examination held on _____

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

We would like to thank our revered Dean, **Dr.P.Thamizhazhagan, M.E., Ph.D.,** for providing infrastructure facilities and whole hearted encouragement for completing our project successfully. Project internal guide

For mostly, we pay our grateful acknowledgement and extend our sincere gratitude to **Dr.L.Jegatha Deborah, M.E., Ph.D.,** Assistant professor and Head, Department of Computer Science and Engineering, University College of Engineering Tindivanam, for extending the facilities of the department towards our project and for unstinting support.

We express our thanks to our guide, **Ms. P. Sathya M.TECH.,** Department of Computer Science and Engineering, University College of Engineering Tindivanam, for guiding us for every phase of the project. We appreciate her thoroughness, tolerance and ability to share her knowledge with us. We thank her for being easily approachable and quite thoughtful. We owe her harnessing our potential and bringing out the best in us. Without her immense support through every step of the way, we could never have it to this extent.

We thank all our teaching and non-teaching faculty members of the department and also our fellow friends for helping us in providing valuable suggestions and timely ideas for the successful completion of the project and also, we extend our immense thanks to our parents and siblings who have been a great source of inspiration and rendered us great support during the course of this project work. We sincerely thank all of them.

ABSTRACT

A cloud-assisted reliable trust computing (CRTC) scheme for data collection in Internet of Things (IoT). The CRTC scheme mainly includes two parts: a reliable approach of obtaining the real data of intelligent sensing devices (ISDs) at a low cost and an effective method of trust computing to evaluate the trustworthiness of ISDs based on the data collected by unmanned aerial vehicles (UAVs). In the proposed method, ISDs fit the forwarding packets information to form inspection information with a small amount of data and then send it to inspection nodes, effectively reducing the cost of routing inspection information. Moreover, we optimize the path planning algorithm for UAVs to reduce flight distance and improve data collection efficiency. The proposed CRTC scheme can effectively identify trustworthy ISDs and improve data collection efficiency.

KEYWORDS: Cloud computing, Internet of Things (IoT), reliable trust computing, security of data collection, unmanned aerial vehicles (UAVs), path planning algorithm.

திட்டப்பணிசுருக்கம்

இன்டர்நெட் ஆஃப் திங்ஸில் (IOT) தரவு சேகரிப்புக்கான கிளவுட்-உதவி நம்பகமான நம்பிக்கைக் கணினி (CRTC) திட்டம். CRTC திட்டம் முக்கியமாக இரண்டு பகுதிகளை உள்ளடக்கியது: குறைந்த செலவில் அறிவார்ந்த உணர்திறன் சாதனங்களின் (ISDகள்) உண்மையான தரவைப் பெறுவதற்கான நம்பகமான அணுகுமுறை மற்றும் வான்வழி வாகனங்கள் மூலம் சேகரிக்கப்பட்ட தரவுகளின் அடிப்படையில் ISD களின் நம்பகத்தன்மையை மதிப்பிடுவதற்கான நம்பிக்கைக் கணினியின் பயனுள்ள முறை (UAV கள்). முன்மொழியப்பட்ட முறையில், ISDகள் பகிர்தல் பாக்கெட்டுகளின் தகவலைப் பொருத்தி, சிறிய அளவிலான தரவுகளுடன் ஆய்வுத் தகவலை உருவாக்கி, பின்னர் அதை ஆய்வு முனைகளுக்கு அனுப்புகிறது, இது ரூட்டிங் ஆய்வுத் தகவலுக்கான செலவைக் குறைக்கிறது. மேலும், விமான தூரத்தைக் குறைக்கவும், தரவு சேகரிப்பு திறனை மேம்படுத்தவும் UAVகளுக்கான பாதை திட்டமிடல் அல்காரிதத்தை மேம்படுத்துகிறோம். முன்மொழியப்பட்ட CRTC திட்டம் நம்பகமான ISDகளை திறம்பட கண்டறிந்து தரவு சேகரிப்பு திறனை மேம்படுத்துகிறது.

முக்கிய வார்த்தைகள்: கிளவுட் கம்ப்யூட்டிங், இன்டர்நெட் ஆஃப் திங்ஸ் (IoT), நம்பகமான நம்பிக்கைக் கம்ப்யூட்டிங், தரவு சேகரிப்பின் பாதுகாப்பு, ஆளில்லா வான்வழி வாகனங்கள் (UAVs), பாதை திட்டமிடல் அல்காரிதம்.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATION

S.NO	ABBREVIATION	DEFINITIONS
1	UAV	Unmanned Aerial Vehicles
2	CRTC	Cloud Reliable Trust Computing
3	IN’s	Inspection Nodes
4	II	Inspection Information
5	ACO	Ant Colony Optimization
6	ISD’s	Intelligent Sensing Devices
7	IOT	Internet Of Things
8	BHA	Black Hole Attack
9	SFA	Selective Forwarding Attack

CHAPTER 1

INTRODUCTION

To ensuring the security of data collection in cloud computing when using intelligent sensing devices (ISDs) in Internet of Things (IoT) applications. The authors propose a cloud-assisted reliable trust computing (CRTC) scheme that uses data fitting to evaluate the trust of nodes and selects nodes with the highest trust as aggregators. The scheme then optimizes the shortest path of an unmanned aerial vehicle (UAV) to collect data from these aggregators.

The CRTC scheme is designed to identify the trust degree of each node, and then select nodes with the highest degree of trust as aggregators. This reduces the probability of data being attacked, thereby improving the security performance of data collection. The main goal of this scheme is to maximize the ratio of packets successfully reaching the aggregators.

The proposed CRTC scheme uses cloud computing to obtain reliable trust values for ISDs, which are often semi connected networks that are restricted by issues such as communication, energy, cost, and accuracy. By leveraging cloud computing resources, this scheme can overcome these limitations and provide a more reliable mechanism for obtaining trustworthy data.

1.1 INTELLIGENT SENSING DEVICES(ISD'S):

ISDs are typically equipped with sensors that can capture various types of data such as temperature, humidity, light, sound, and motion. These sensors can be embedded in various IoT devices such as smart homes, wearables, and industrial equipment to collect data that can be used for various applications. The CRTC scheme proposed in the paper uses a reliable approach to obtain real data from ISDs at a low cost. The ISDs fit the forwarding packets information to form inspection information (II) with a small amount of data and then send II to inspection nodes. This effectively reduces the cost of routing II and ensures that only trustworthy nodes are selected as aggregators for data collection.

1.2 CLOUD ASSISTED RELIABLE TRUST COMPUTING(CRTC)

The Cloud-assisted Reliable Trust Computing (CRTC) scheme is a proposed approach for ensuring secure and trustworthy data collection in Internet of Things (IoT) applications. The CRTC scheme uses intelligent sensing devices (ISDs) to collect data, which is then evaluated for trustworthiness based on a reliable approach that reduces the cost of routing inspection information (II). The scheme also uses unmanned aerial vehicles (UAVs) to collect data and evaluate the trustworthiness of ISDs.The CRTC scheme comprehensively considers the complete trust of each node and the stability of that trust over multiple rounds of data transmission. It selects nodes with the highest degree of trust as aggregators, which reduces the probability of data being attacked and improves the security performance of data collection.

1.3 UNMANNED AERIAL VEHICLES(UAV)

Unmanned aerial vehicles (UAVs) play a critical role in collecting data and evaluating the trustworthiness of intelligent sensing devices (ISDs). The UAVs are equipped with perception functions that enable them to collect data from ISDs and transmit it to the cloud for analysis. The UAVs in the CRTC scheme is to collect data from ISDs and evaluate its trustworthiness based on a reliable approach. This helps ensure secure and trustworthy data collection in IoT applications.

1.4 CLOUD COMPUTING

The cloud provides a powerful computing platform with flexible and dynamic services that are well-suited for smart grid applications. The cloud server can process and analyze massive data from ISDs, which improves the reliability and accuracy of IoT applications. The CRTC scheme also uses the cloud to host a data center where collected data is stored and analyzed. The cloud provides a secure environment for storing sensitive data and ensures that only authorized users have access to it.

CHAPTER 2

LITERATURE SURVEY

2.1 X. Zhang and Z. Ge, "Local parameter optimization of LSSVM for Industrial soft sensing with big data and cloud implementation," IEEE Trans. Ind. Informat., vol. 16, no. 5, pp. 2917–2928, May 2020

The paper proposes a cloud-assisted reliable trust computing scheme for data collection in the Internet of Things (IoT). The scheme aims to address the issue of data reliability and trustworthiness in IoT by combining blockchain technology and cloud computing. The proposed scheme consists of three stages: data collection, data verification, and data storage. In the data collection stage, data is collected from IoT devices and transmitted to the cloud. In the data verification stage, the cloud verifies the authenticity and integrity of the data using a trust computing module. The trust computing module assigns a trust value to each device based on its behavior history and other relevant factors. The data is then stored in the blockchain-based distributed storage system in the data storage stage.The proposed scheme was evaluated using simulation experiments, and the results showed that it achieved high reliability and trustworthiness for data collection in IoT.

Merit: High reliability: The scheme achieves high reliability in data collection by using a trust computing module that assigns a trust value to each device based on its behavior history and other relevant factors. This ensures that only trustworthy devices are used to collect data**.**

Demerit: Centralization: The scheme relies on cloud computing resources for data verification and storage, which can lead to centralization and create a single point of failure. Any disruption or failure in the cloud infrastructure could result in the loss of data or system downtime.

2.2 J. Huang, L. Kong, G. Chen, M. Wu, X. Liu, and P. Zeng, "Towards secure industrial IoT: Blockchain system with credit-based consensus mechanism," IEEE Trans. Ind. Informat., vol. 15, no. 6, pp. 3680–3689, Jun. 2019

The paper "A Cloud-Assisted Reliable Trust Computing Scheme for Data Collection in Internet of Things" proposes a novel scheme to ensure secure data collection in the Internet of Things (IoT) environment. The authors highlight that the IoT paradigm is facing significant challenges in terms of trust, reliability, and security due to the vast number of devices connected to the network, as well as the sensitivity of the data they collect.

Merit: Firstly, the paper proposes a novel blockchain-based system that ensures secure communication and data exchange in Internet of Things (IoT) environments. The system utilizes a credit-based consensus mechanism that reduces the computational requirements and enhances the scalability of the network.

Demerit: The paper appears to provide valuable contributions towards securing IoT environments using a novel blockchain-based system with a credit-based consensus mechanism.

2.3 M. Shen, A. Liu, G. Huang, N. Xiong, and H. Lu, "ATTDC: An active and Trace-able trust data collection scheme for industrial security in smart cities," IEEE Internet Things J., vol. 8, no. 8, pp. 6437–6453, Apr. 2021

The ATTDC scheme provides enhanced security for industrial systems by actively collecting and tracing data to ensure that only trusted entities are granted access. This helps to prevent unauthorized access and potential security breaches..The ATTDC scheme provides real-time monitoring of industrial

systems, which can help identify potential security threats and vulnerabilities. This enables proactive measures to be taken to prevent security breaches.

Merit: The ATTDC scheme provides enhanced security for industrial systems by actively collecting and tracing data to ensure that only trusted entities are granted access. This helps to prevent unauthorized access and potential security breaches.

Demerit: The implementation of the ATTDC scheme may require significant investment in hardware and software infrastructure, as well as ongoing maintenance and support.

2.4 P. Mishra, E. S. Pilli, V. Varadharajan, and U. Tupakula, "Intrusion detection techniques in cloud environment: A survey." J. Netw. Comput. Appl.,vol. 77, pp. 18–47, 2017

The paper discusses different types of intrusion detection techniques, including signature-based, anomaly-based, and hybrid approaches. A study by Liu et al. (2019) noted that hybrid approaches can enhance the accuracy and efficiency of intrusion detection in cloud environments.The paper highlights some of the challenges associated with intrusion detection in cloud environments, including the dynamic and distributed nature of cloud infrastructure, the diversity of cloud services, and the volume and complexity of cloud data. A study by Alqahtani et al. (2021) noted that these challenges can make intrusion detection more difficult in cloud environments.

Merit: The paper provides a comprehensive overview of intrusion detection techniques in cloud computing environments, which can be helpful for researchers and practitioners who are interested in this field.

Demerit: The paper was published in 2017, and as such, it may not reflect the most recent developments in intrusion detection techniques in cloud environments. More recent studies may provide more up-to-date information on this topic.

2.5 Y. Liu, M. Dong, K. Ota, and A. Liu, "ActiveTrust: Secure and trustable routing in wireless sensor networks," IEEE Trans. Inf. Forensics Secur., vol. 11, no. 9, pp. 2013–2027, Sep. 2016.

The paper discusses the problem of secure and reliable routing in wireless sensor networks (WSNs), which are vulnerable to attacks due to their distributed and resource-constrained nature. The authors propose a new routing protocol called Active Trust that addresses these challenges by providing both security and trust features. Active Trust uses a combination of cryptographic techniques and trust-based mechanisms to ensure the integrity and authenticity of the data transmitted in the network

Merit: ActiveTrust uses digital signatures and message authentication codes to ensure data integrity and authenticity, which helps prevent unauthorized modifications and impersonations..

Demerit: The use of cryptographic techniques and trust-based mechanisms can make the protocol more complex than other routing protocols, which may make it more difficult to implement and maintain.

2.6 C. Huang, G. Huang, W. Liu, R. Wang, and M. Xie, "A parallel joint optimized relay selection protocol for wake-up radio enabled WSNs," Phys. Commun., vol. 47, Aug. 2021

The paper addresses the problem of energy consumption in wireless sensor networks (WSNs), which can significantly impact the network lifetime. The authors propose a new relay selection protocol called P-JORS, which aims to minimize energy consumption by jointly optimizing the relay selection and wake-up scheduling.

Merit: P-JORS is designed to minimize energy consumption in WSNs by jointly optimizing relay selection and wake-up scheduling, which leads to increased energy efficiency.

Demerit: The additional wake-up scheduling and relay selection mechanisms can increase communication overhead, which could impact the performance of the network.

2.7 D. C. Mehetre, S. E. Roslin, and S. J. Wagh, "Detection and prevention of black hole and selective forwarding attack in clustered WSN with active trust," Cluster Compute., vol. 22, no. 1, pp. 1313–1328, 2019

as environmental monitoring, healthcare, and smart cities. However, due to the open nature of wireless communication, WSNs are vulnerable to various attacks such as black Wireless Sensor Networks (WSNs) are widely used in various applications such hole and selective forwarding attacks. In this paper, we propose a new approach for detecting and preventing these attacks in clustered WSNs using active trust.

Merit: The proposed approach provides an effective way to detect and prevent black hole and selective forwarding attacks in clustered WSNs using active trust.
Demerit: The authors evaluated the proposed approach through simulations, but further evaluation in real-world scenarios is needed to validate its effectiveness and practicality.

CHAPTER 3

PROBLEM STATEMENT

- To maximize the success rate of data collection while ensuring security performance. $W = M_{rec} / M_{sen}$
- Maximize the ratio T_n of node trust verification. $Tn = N_{obt} / N_{tot.}$
- Maximize the V_{trust} speed of trust acquisition. $V_{trust} = 1/T_{uav}$

3.1 OBJECTIVES

- To evaluate the trustworthiness of nodes in IoT networks using data fitting techniques.
- To select nodes with the highest trust scores as aggregators to minimize the probability of data being attacked and ensure high security performance.
- To optimize the shortest path for a UAV to collect data from aggregators, which reduces time and cost required for manual collection.
- To use a cloud server for further processing and analysis of collected data, which enables efficient and timely data collection from multiple aggregators located in different areas.
- To propose a reliable trust computing scheme that can detect and prevent black hole attacks, selective forwarding attacks, and other malicious activities in IoT networks.

3.2 SCOPE

- High secure data collection
- Path optimization
- Preventing malicious attack
- Reliable trust computing

3.3 SYSTEM DESCRIPTION

The Internet of Things (IoT) describes the network of physical objects that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet. Routing protocols is a mechanism of detecting best route path from single node to multiple nodes or signal node in order to transfer data between network and it specify how routing techniques in routers communicate between routers to accomplish communication tasks. The routing table must be secure before transmitted a data packet. To detect and prevent the malicious attack like black hole attack, Selective forwarding attack and clone attack. And check the transmitted data packet is same as original packet. Calculate the trust evaluation of received packet with the help of aggregators nodes of past performance.

CHAPTER 4
REQUIREMENT SPECIFICATION

SYSTEM REQUIREMENT

The requirement analysis is the technical requirement of the software product. It is the first step in the requirement analysis process. The requirements also provide useful constraints from a user, an operational and administrative perspective. It is a parameter which describes its user interface, hardware and software requirements. The requirement analysis has two factors

- Functional requirements.
- Non-functional requirements.

4.1 FUNCTIONAL REQUIREMENTS

A functional requirement is a declaration of the intended function of a system and its components. The functional requirements include data collection, trust evaluation, aggregator selection, UAV path optimization, cloud-based processing and analysis, as well as malicious activity detection and prevention.

4.1.1 Hardware requirements

Processor	- Intel's Xeon-D (D-1700 and D-2700)
Speed	- 4 GHz
Ram	- 8 GB and above
Hard Disk	- 1 TB

4.1.2 Software requirements

Operating System	- Windows or Linux
Programming language	- Python
IDE	- Visual Studio Code or Sublime text.

4.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements or NFRs are a set of specifications that describe the system's operation capabilities and constraints and attempt to improve its functionality. These are basically the requirements that outline how well it will operate including things like speed, security, reliability, data integrity, etc.

Scalability

- The capability of a system, network or process to handle a growing amount of work.

Reliability

- To quantify the probability that a system performs its intended function correctly throughout its mission time, or its complement value.

Performance Characteristics

- Speed, throughput and execution times depends on the size of the input dataset.

Security

- Provide security of the code from unknown uninvited guest.

Flexibility

- Supports easy adaptation of the system to meet future requirements and changes through configuration, rather than new development.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

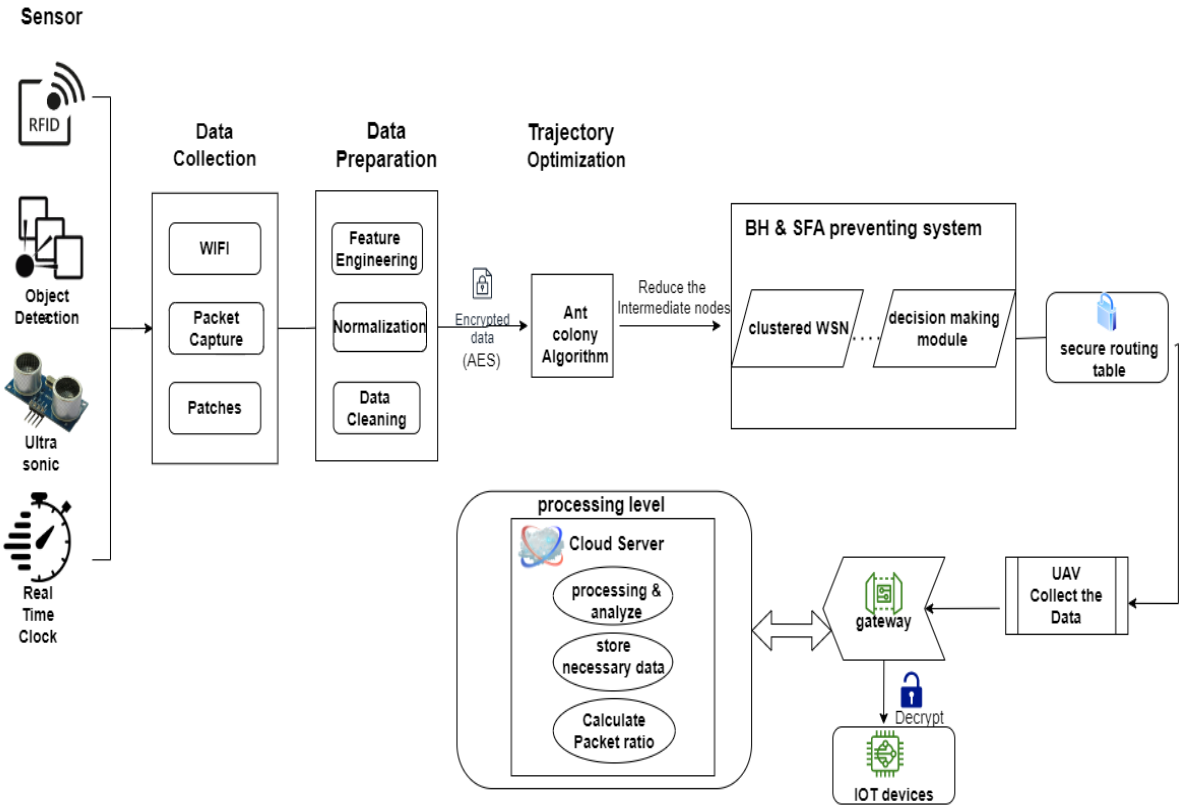


Fig 5.1 System architecture

5.2 MODULE DESCRIPTION

5.2.1 Data Collection

This module is responsible for collecting data from intelligent sensing devices (ISDs) deployed in the industrial environment. The module may include sub-modules for data preprocessing and filtering.

Communication Module: This module enables communication between ISDs and the cloud server. It may include sub-modules for data transmission, encryption/decryption, and error correction.

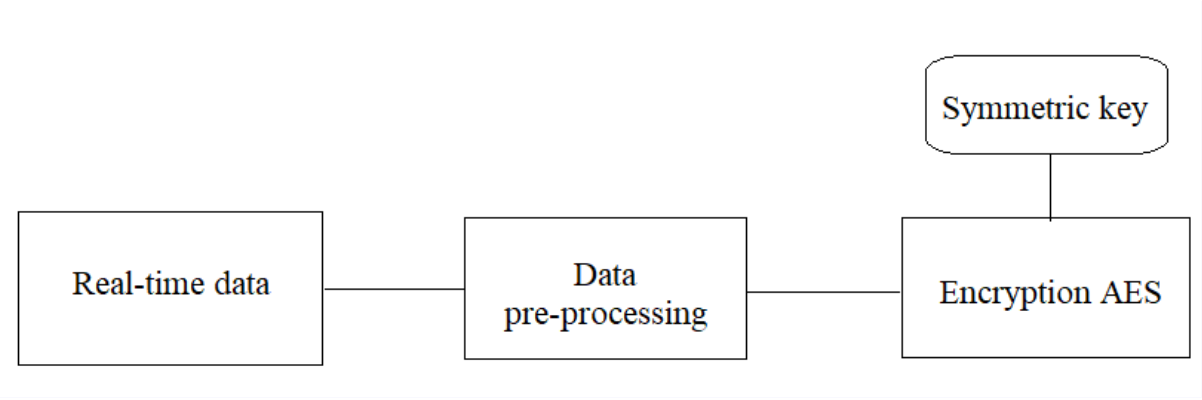


Fig 5.2.1 AES Encryption

Algorithm (For AES Encryption)

1. state = M

2. Add Round Key(state, &w[0])

3. for i = 1 step 1 to9

4. SubBytes (state)

5. Shift Rows(state)

6. MixColumns(state)

7. Add RoundKey(state, &w[i*4])

8. end for

9. SubBytes(state)

10.Shift Rows (state)

11.Add RoundKey(state, &w[40])

5.2.2 Trust Computing Module

This module evaluates the reliability of each ISD based on its historical behavior and other contextual information such as location, time, and environmental conditions. The trust score assigned to each ISD determines its level of access to the cloud server and influences its contribution to decision-making processes Once the reputation score is calculated for each ISD, it is used to determine its level of access to the cloud server and influence its contribution to decision-making processes. For example, an ISD with a high reputation score may be given priority access to the cloud server or have a greater influence on decision-making processes than an ISD with a low reputation score.

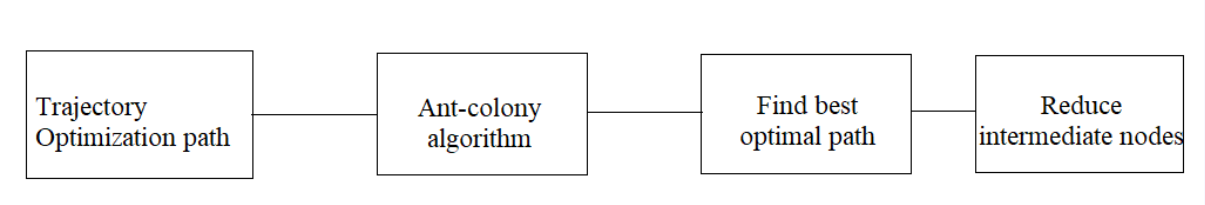


Fig 5.2.2 Optimization path

Algorithm (For Shortest path)

Input: α , β, ρ , γ, δ, τ0 , Imax; dij

Output: Ln, L_length

1: For each edge(i, j)

2: τ (i, j) = τ0

3: End for

4: While (n = 1; n + +; n $\leq$ Imax)

5: For each j $\in$ SV //SV is the set of all nodes

6: The UAV randomly selects a node Vk to start

7: Ωj = Ωj − Vk // Ωj is the set of unvisited nodes

8: Pcur,j = Vk // Pcur,j is the current position at the node Vk

9: Calculate the visited probability of nodes using (8)

10: Select node Vt as the next visit node of the UAV

11: Ln = Ln $\cup$ (Vk, Vt) //Add edge (Vk, Vt) to path Ln

12: Compute energy consumption of node Vt and flight cost of the UAV according to (11) and (12)

13: Update the τ (k, t)of edge(Vk, Vt) using (14)

14: End for

15: Compute Llength // trajectory length for one iteration

16: End while

17: Compare Llength after Imax iterations

18: Llength = Llength

19: Return Ln and Llength

Algorithm (For add Nearby Nodes as Inspection Nodes.)

Input: Ln, SCHs:set of INs, SV :set of all nodes

Output: Lnew n

1: For each path (Vi, Vj) ∈ Ln

2: If there is a node Vξ ∈ SV and dVξ,uav ≤ C when UAV passes through the path (Vi, Vj) then

3: SCHs = SCHs ∪ Vξ

4: End if

5: Lnew n = Ln ∪ (Vi, Vξ) ∪ (Vξ, Vj)

6: End for

7: Return Lnew

Algorithm (For Reduce intermediate nodes)

Input: Ln, SCHs

Output: Lbest n

1: For paths Vi−1 → Vi → Vi+1 ∈ Lold n //Vi−1, Vi, Vi+1 is the sequentially visited node in set SCHs

2: If (Vi−1, Vi) + D(Vi, Vi+1) D(Vi−1, Vi+1) Then

3: SCHs = SCHs − Vi

4: Path (Vi−1→Vi → Vi+1) ∈ L / old n

5: Remove (Vi−1→Vi → Vi) and add path (Vi−1 → Vi+1)

6: If there is a node Vt between path (Vi−1 → Vi+1) Then

7: Combine algorithm 2 to add an inspection node Vt

8: SCHs = SCHs + Vt

9: Lbest n = Ln ∪ (Vi−1, Vt) ∪ (Vi+1, Vj)

10: else if

11: Lbest n = Ln ∪ (Vi−1, Vi+1)

12: End if

13: End if

14: End for

15: Return Lbest

5.2.3 Cloud Computing Module

This module provides cloud-based processing power and storage capacity to process and analyze data collected from ISDs. It may include sub-modules for data storage, processing, and analysis. Once the data is collected from ISDs, it is transmitted to the cloud server for processing and analysis. The cloud server has strong computing power and a large storage capacity, which enables it to process and analyze massive amounts of data from ISDs.

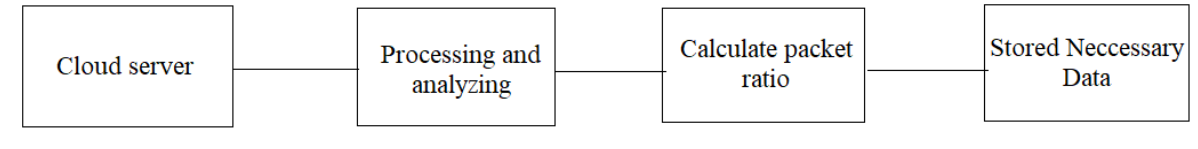


Fig 5.2.3 Storage and processing

5.2.4 IOT devices

The analyzed data can then be transmitted back to IoT devices for decision-making processes or used to generate feedback for users through a graphical user interface (GUI). For example, in a smart grid application, the analyzed data can be used to optimize energy consumption or detect anomalies in the grid. The analyzed data can then be used to improve decision-making processes or generate feedback for users through a graphical user interface (GUI).

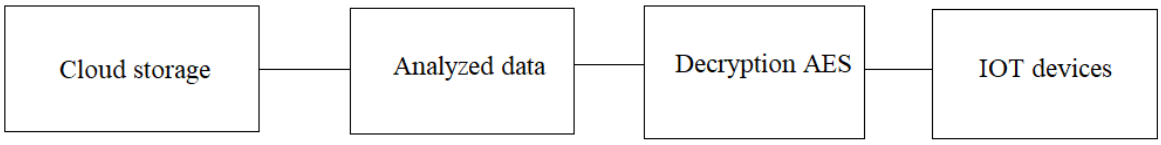


Fig 5.2.4 IOT devices

Algorithm (For decryption AES)

1.decrypt(algorithm, cipherText, key, iv):

2.cipher = Cipher.getInstance(algorithm)

3.cipher.init(Cipher.DECRYPT_MODE, key, iv)

4.decodedCipherText = base64Decode(cipherText)

5.plainText = cipher.doFinal(decodedCipherText)

6.return byteArrayToString(plainText)

The proposed system architecture aims to improve the efficiency and reliability of data collection in IoT applications by utilizing cloud-based processing power and implementing a trust computing scheme that ensures secure and reliable data transmission from ISDs.

5.3 UML DIAGRAMS

5.3.1 Use case Diagram

Use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a user analysis. Its purpose is to prevent a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases) and any dependencies between those use cases. The main purpose of a use case diagram is to show that system functions are performed from actor.

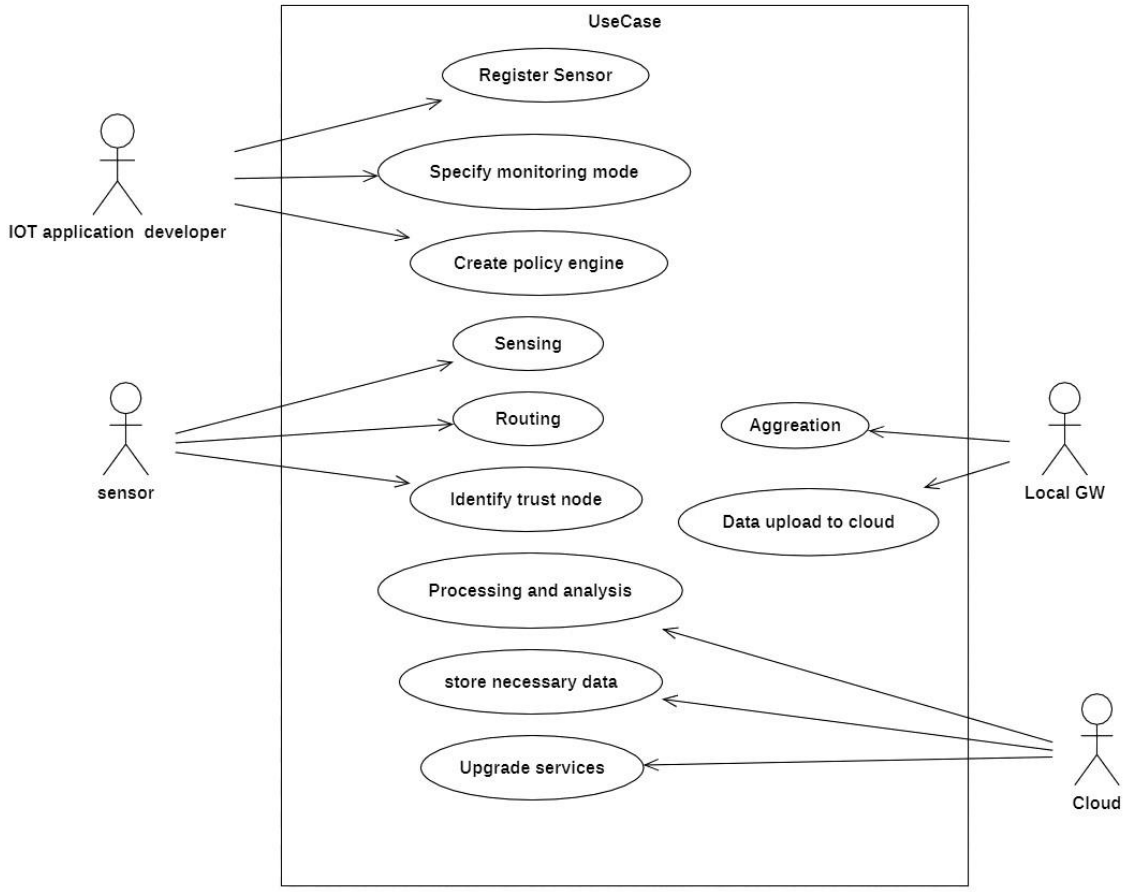


Fig 5.3.1 Use Case Diagram

5.3.2 Class Diagram

A class diagram in the UML is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes and relationship between the classes. In software engineering, a class diagram in the unified modeling language (UML) is a type of statistic structure diagram that describes the structure of a system's classes, their attributes, operation (or methods) and the relationship among the classes. The class diagram gives detailed look about the attribute and operations of each class.

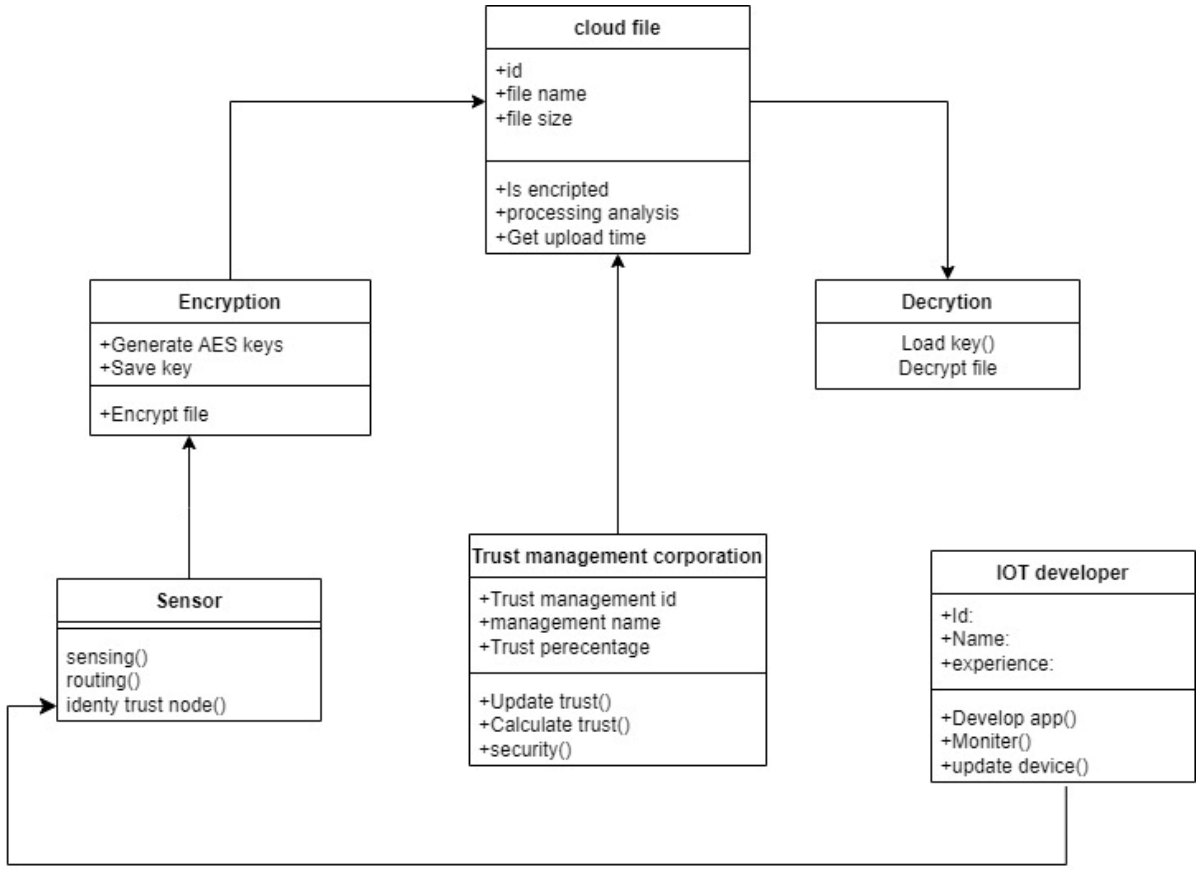


Fig 5.3.2 Class diagram

5.3.3 Sequence Diagram

Sequence diagrams are type of interaction diagrams. This describes interactions among classes. These interactions are modeled as exchanged of messages. Sequence diagram gives detailed interaction of the object**.**

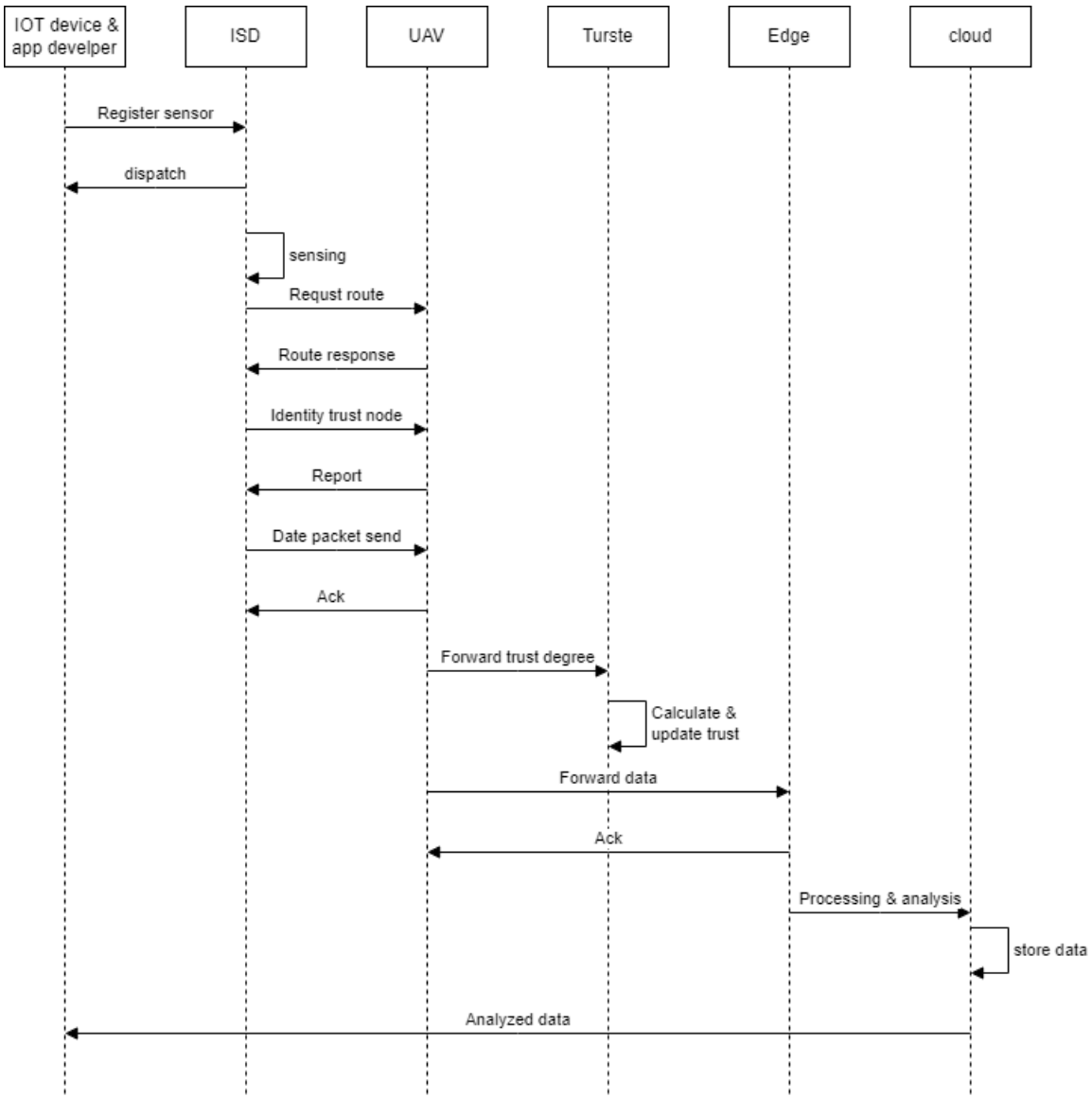


Fig 5.3.3 Sequence diagram

5.3.4 Activity Diagram

An activity diagram provides a view of the behavior of a system by describing the sequence of actions in a process.

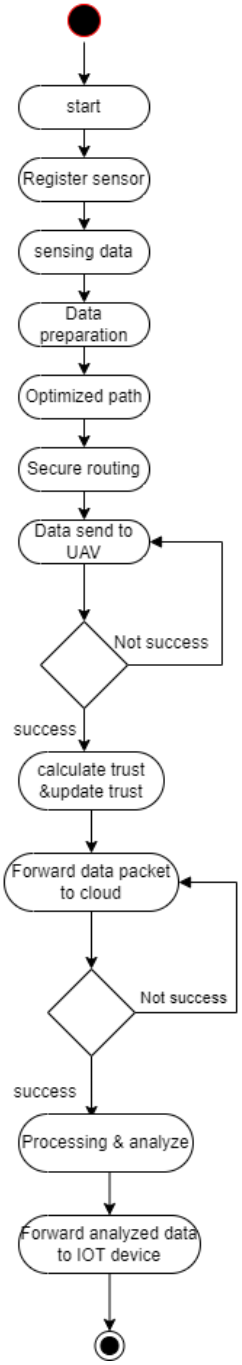


Fig 5.3.4 Activity diagram

5.3.5 State Chart Diagram

State machine diagrams are sometimes known as state diagrams or state chart diagrams as well. These are very useful to describe the behavior of objects that act differently according to the state they are in at that moment. State machine diagrams are sometimes known as state diagrams or state chart diagrams as *F*

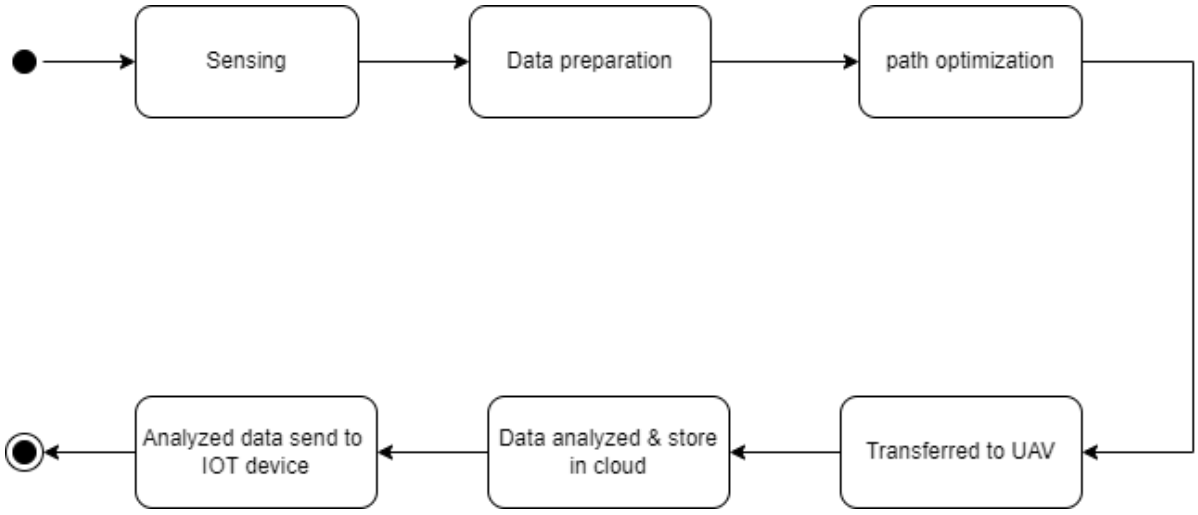


Fig 5.3.5 State chart diagram

5.3.6 Deployment Diagram

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them. Deployment diagrams are typically used to visualize the physical hardware and software of a system Deployment diagrams help model the hardware topology of a system compared to other UML diagram types which mostly outline the logical components of a system.

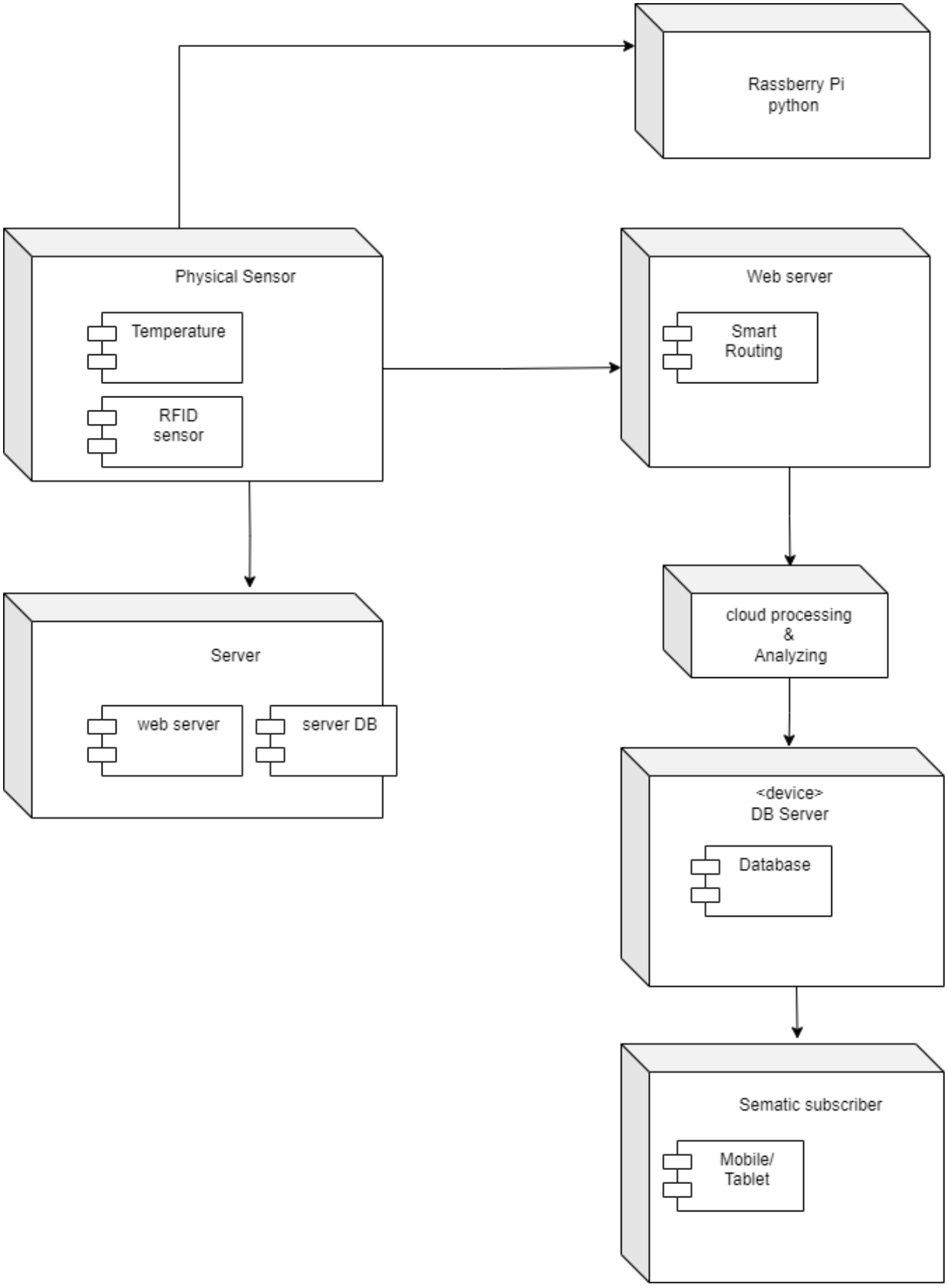


Fig 5.3.6 Deployment diagram

5.4 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the 'flow' of a data through an information system. It differs from the flowchart as it shows the data instead of the control flow of the program. A data flow diagram can also be used for the visualization of data processing. Data flow diagram is designed to show how the system is divided onto the smaller portions and to highlight the flow of data between those parts.

DFD is an important technique for modeling a systems high level detail by showing how input data is transformed to output results through a sequence functional transformation. DFD consists of four major components: entities, processes, data stores and data flow.

5.4.1 DFD-Level 0

A context diagram is a top level (also known as "Level 0") data flow diagram. It only contains one process node ("Process 0") that generalizes the function of the entire system in relationship to external entities. DFD diagrams can be made in several nested layers. It is the most basic form of DFD. There is only one process in the system and all the data flows either into or out of this process. Context level DFD demonstrates the interactions between the process and external entities.

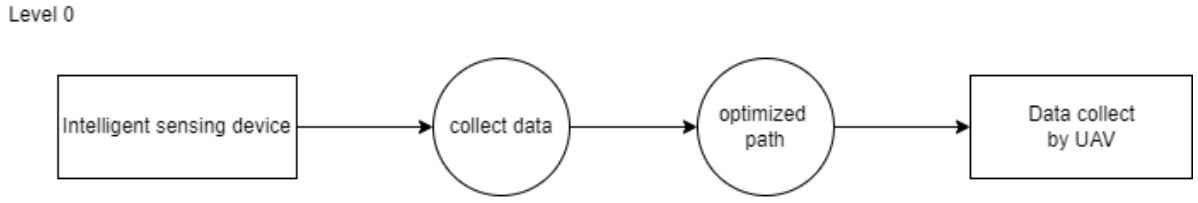


Fig 5.4.1 DFD level 0

5.4.2 DFD-Level 1

DFD-level 1 is more detailed than a DFD-level 0 but not as detailed as a DFD-level 2. It breaks down the main processes into sub processes that can then be analyzed and improved on a more intimate level.

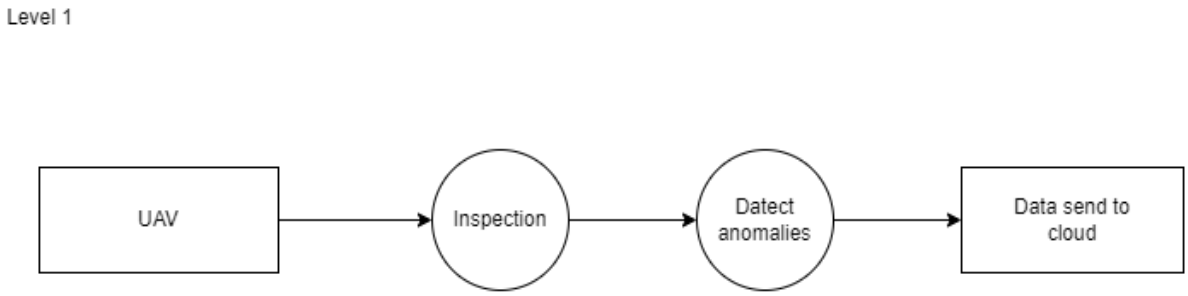


Fig 5.4.2 DFD level 1

5.4.3 DFD-Level 2

A DFD-level 2 offers a more detailed look at the processes that make up an information system than a DFD-level 1 does. It can be used to plan or record the specific makeup of a system**.**

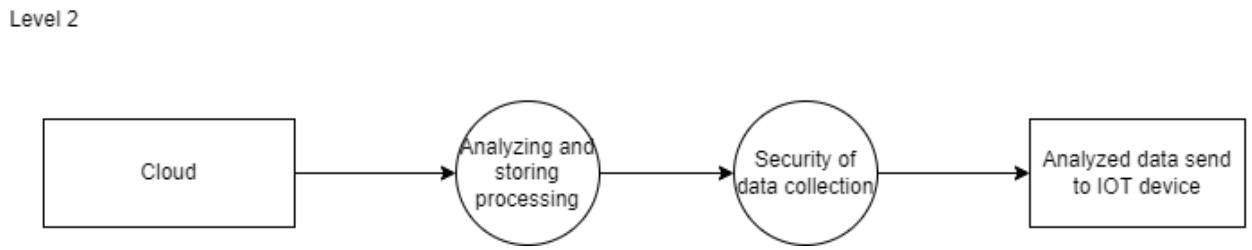


Fig 5.4.3 DFD level 2

CHAPTER 6

IMPLEMENTATION

6.1 Technologies used

The project can be done successfully by using various technologies for different function or operation.

6.1.1 Python

Python is used for data analysis or machine learning tasks related to evaluating trust levels of nodes in the network, but this is not explicitly stated in the document. Python are usually used to quickly and accurately obtain the fitting results. Python's strength lies in its extensive library ecosystem.

Libraries like NumPy, pandas, Matplotlib, and scikit-learn are widely used for data analysis and machine learning tasks. Python has a wide range of frameworks that provide pre-built libraries, tools, and templates to simplify and expedite the development process for specific applications. Pyramid is a flexible and scalable web framework that follows the model-view-controller (MVC) architectural pattern. It is designed to be adaptable to different project sizes and requirements. Pyramid offers a variety of features for building web applications, including URL routing, template rendering, authentication, and database integration. It emphasizes flexibility and allows developers to make choices regarding the project's structure and components.

6.1.2 Optimized Trajectory Path

UAV Path Optimization discusses the optimization of trajectory paths for unmanned aerial vehicles (UAVs) using an ant colony algorithm and 2-OPT technology. A method to optimize the route of UAVs by adding intermediate nodes (INs) to reduce the burden of INs and balance network energy consumption. optimizing trajectory paths for UAVs is an important task in ensuring efficient and effective operation of these devices. The authors' proposed method provides a way to do this by leveraging ant colony algorithms and 2-OPT technology to find an optimal route with minimal energy consumption and balanced network load.

6.1.3 Cloud Server

A cloud server, also known as a virtual server or virtual machine (VM). is a computing resource that is provisioned and hosted in a cloud computing environment. Instead of running on physical hardware, a cloud server operates on virtualized infrastructure provided by a cloud service provider. Popular cloud service providers offering cloud server solutions include Amazon Web Services (AWS). Microsoft Azure, Google Cloud Platform (GCP), and IBM Cloud, among others. These providers offer various instance types, operating systems, and configurations to cater to different workload requirements.

6.2 Snapshots

Dataset for 10 nodes

600 600

110.0 225.0

161.0 280.0

325.0 554.0

490.0 285.0

157.0 443.0

283.0 379.0

397.0 566.0

306.0 360.0

343.0 110.0

552.0 199.0

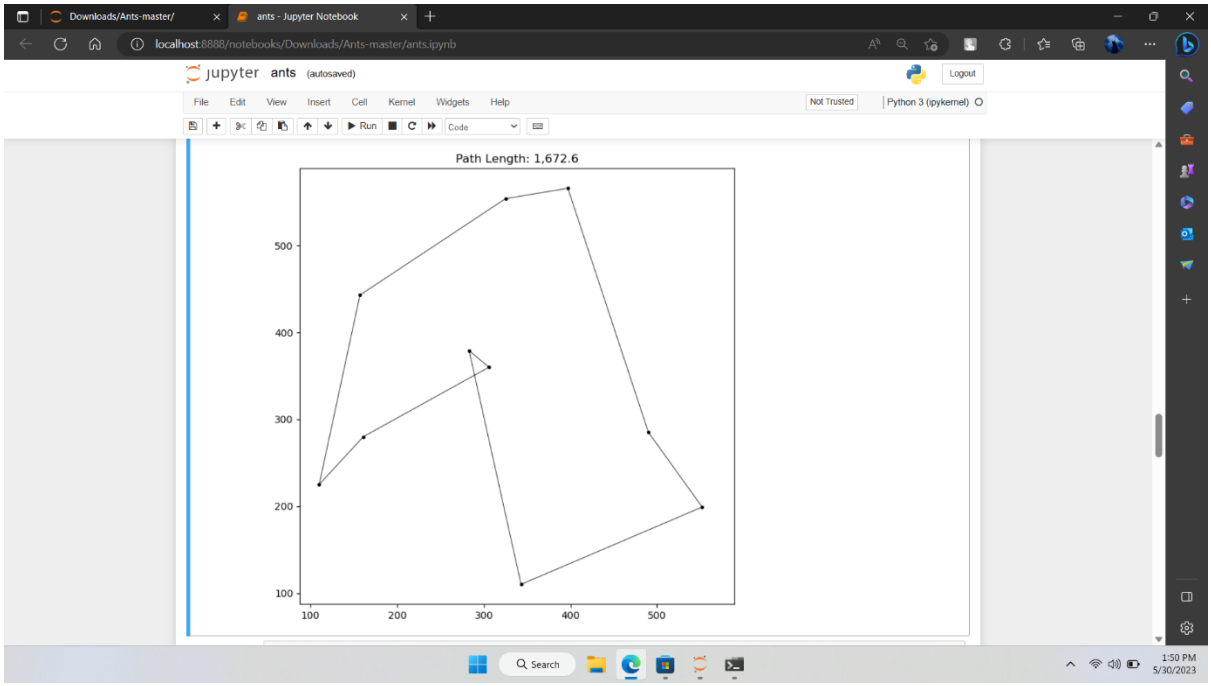


Fig 6.1 Finding path for 10 nodes

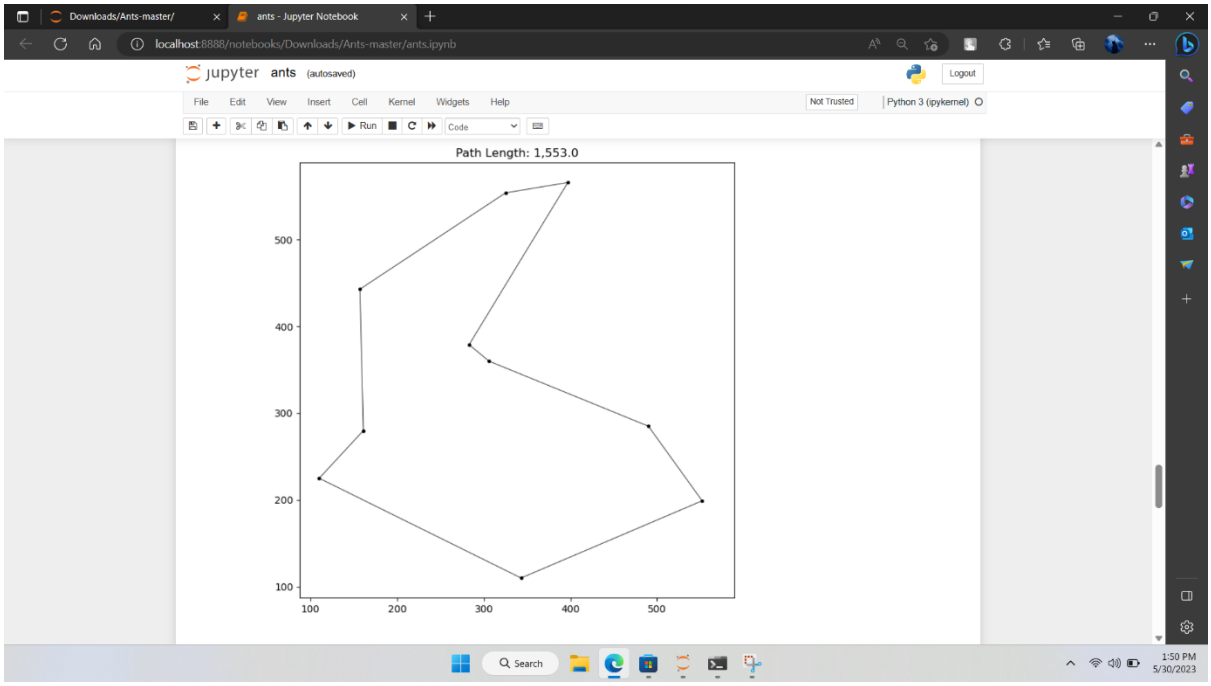


Fig 6.2 Optimal path for 10 nodes

Dataset for 100 nodes

600 600	221.3036 324.1944	51.3431 442.4751
273.1588 72.6027	464.6742 403.1801	138.0119 393.6293
423.7493 428.0372	549.3704 134.9750	427.1514 411.3751
509.4693 426.8362	10.0000 346.6937	589.0593 215.0038
512.0536 520.2136	152.5254 450.5733	493.3952 289.8690
112.1935 453.8568	257.7839 341.6199	438.2191 334.2714
54.4830 231.6509	354.1409 149.3780	392.4814 357.9693
266.2236 220.0501	371.5071 590.0000	58.8089 181.2302
168.0002 365.8709	430.2164 347.7012	311.7615 353.9954
180.0541 359.8832	428.4436 526.3642	165.5282 248.1620
87.0681 286.3926	384.8782 133.7609	130.9518 202.2638
315.6692 103.6946	246.6101 409.2352	261.5480 383.9406
457.7327 520.4371	53.7277 276.0534	165.5969 315.6168

595.9071 303.7975 330.1640 499.7318 40.6437 429.6571

507.6590 368.8210 516.7853 226.8680 266.8353 10.0000

56.6928 367.5756 534.9506 337.7491 86.7809 336.4725

394.4477 513.8520 186.7959 494.8372 468.0013 344.5027

188.3190 445.5601 297.4665 259.6835 171.3398 542.6068

211.9338 223.0796 339.0843 125.1764 578.7719 403.2444

513.5830 234.8127 456.5466 279.3881 539.0705 180.1515

256.7040 513.9357 301.4429 341.7011 335.8320 362.5449

195.8223 110.9538 241.7536 551.8405 416.6955 512.8015

379.3163 268.0682 504.5441 194.1188 313.8028 257.0729

419.0551 230.5192 300.6439 163.8465 322.4235 295.7811

382.4749 174.4978 559.7077 243.4871 233.9381 419.6163

523.4022 491.6742 542.6786 164.3521 260.6304 462.8601

171.4522 31.8651 99.5465 367.5969 49.2269 445.3535

168.9303 293.8379 404.9224 278.8069 208.4506 270.0458

97.2369 149.8287 195.4291 96.0121 466.2847 260.2540

24.3200 366.1980 361.0886 415.9177 575.2262 269.1700

58.4656 232.0548 233.4325 74.1015 349.3280 109.7422

147.4940 473.6869 343.2292 338.0936 543.8085 328.3337

458.7439 267.5501 484.3626 122.6139 396.3891 185.0917

276.8106 51.5341 81.8307 430.1109

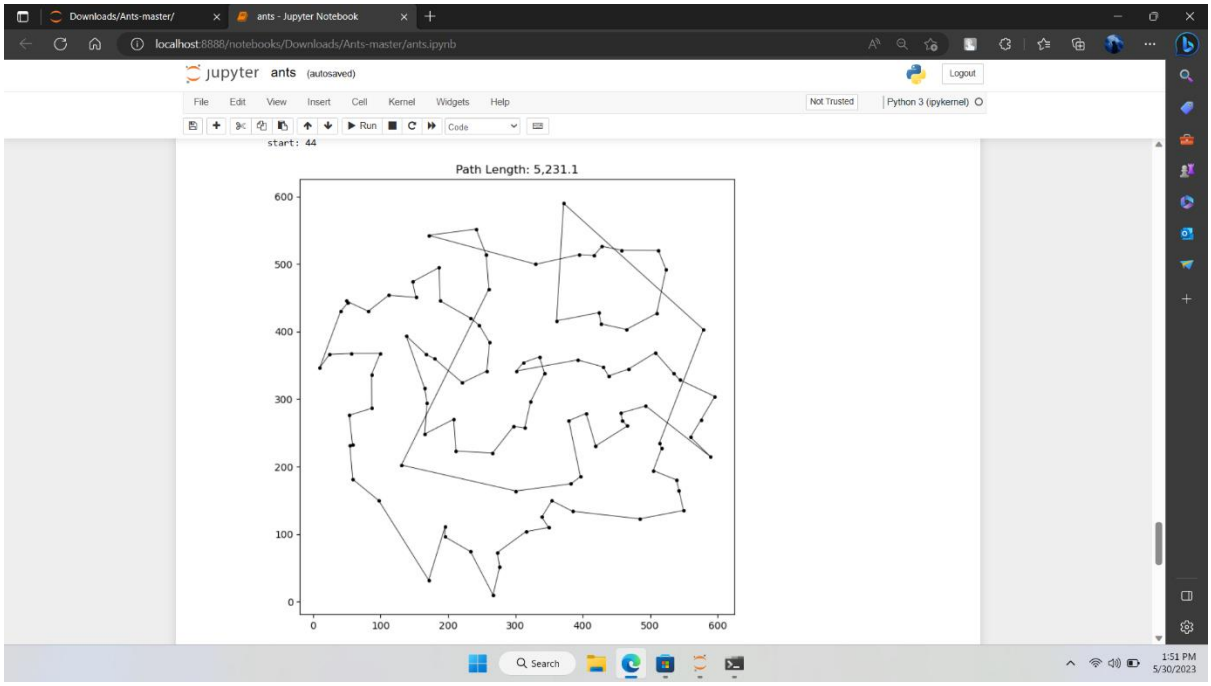


Fig 6.3 Finding path for 100 nodes

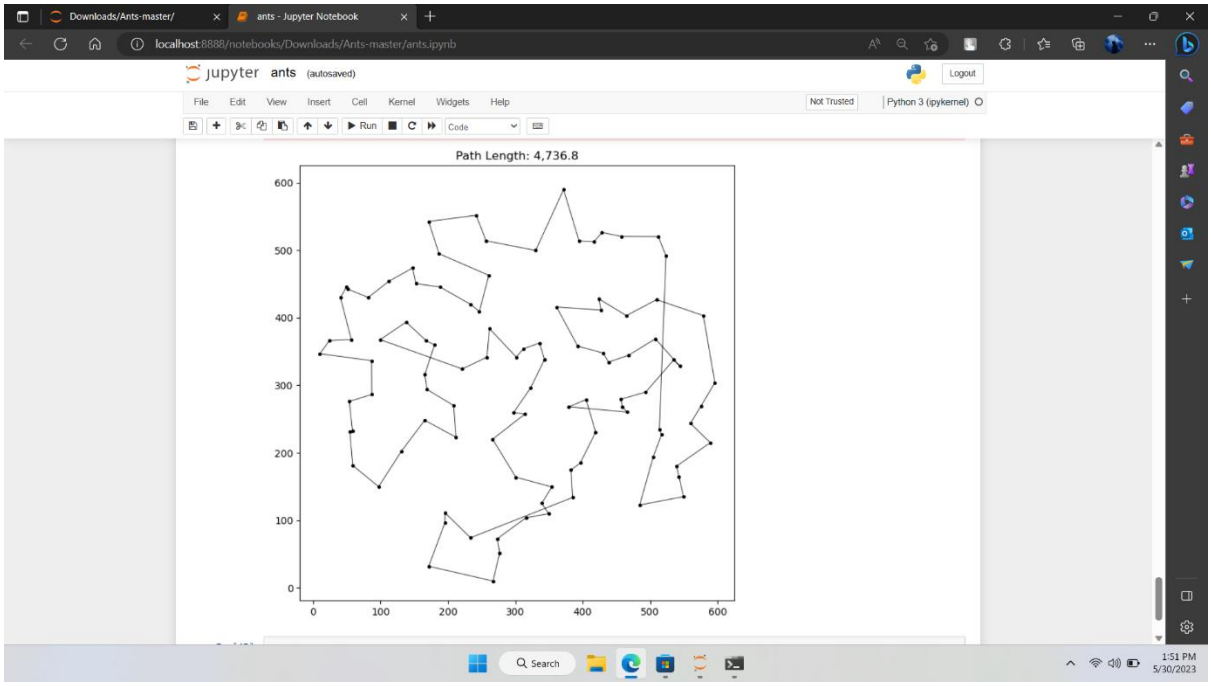


Fig 6.4 Optimal path for 100 nodes

CHAPTER 7

TESTING

7.1 TESTING OBJECTIVES

Testing is performed to identify errors. A good test case is one that has a high probability of finding an undiscovered error. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example, the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases: assemblies and/or a finished product. It is the process of exercising software.

- Static analysis is used to investigate the structural properties of the source.
- Dynamic testing is used to investigate the behavior of the source code.

Static testing is a software testing method that examines a program -- along with any associated documents -- but does not require the program to be executed. Dynamic testing, the other main category of software testing, requires testers to interact with the program while it runs.

7.2 TYPES OF TESTING

7.2.1 Unit Testing

Unit testing involves the design of test cases that validate that the Internal program logic is functioning properly, and that program input produces valid output. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the 38 completions of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined input and expected results.

7.2.2 Integration Testing

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the setoff modules which make up the project. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have

passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs.

The new module and its interring communications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

7.2.3 Functional Testing

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values, such as logically related inputs, files of identical elements, and empty files. Three type of test in functional test

- Performance Test
- Stress Test
- Structure Test
- System Test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items.

- Valid Input: identified classes of valid input must be accepted.
- Invalid Input: identified classes of invalid input must be rejected.
- Functions: identified functions must be exercised.
- Output: identified classes of application outputs must be exercised.
- Systems/Procedures: interfacing systems or procedures must be invoked.

7.2.3.1 Performance Testing

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit. The Performance test ensures that the output is produced within the

time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results, the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

7.2.3.2 Stress Testing

Stress test is the test designed to intentionally break the unit. A great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

7.2.3.3 Structure Testing

Structure tests are concerned with exercising the internal logic of a program and traversing particular execution path. The way in which White-Box test strategy was employed to ensure that the test cases could guarantee that all independent paths within a module have been exercise at least once.

- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.

Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs.

7.2.3.4 System Testing

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability if finding an as-yet-undiscovered error. A successful test is one that uncovers an as-yet-undiscovered error. System testing is the stage of implementation which is and steps for run program, string, system and is important in adopting a successful new system.

This is the last chance to detect and correct errors before the system is installed foe user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise, the program or the project is not said to be complete. Software testing is the critical element of software quality coding. Testing is the process of executing the program with the intent of finding the error. A successful test is one that undiscovered error. A successful test is one that uncovers a yet undiscovered error. Any engineering product can be tested in one of the two ways.

7.3 TESTING TECHNIQUES

7.3.1 Testing

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as- yet undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing. The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Testing is the process of executing the program with the intent of finding the errors. A good test case design is one that has a high probability of finding the errors. A successful test is one that uncovers

an undiscovered error. Any engineering product can be tested in one of the following two ways:

7.3.2 White Box Testing

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

- Flow graph notation
- Cyclometric complexity

7.3.3 Black Box Testing

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software. It focuses on ensuring that the system behaves correctly from a user's perspective and meets the specified requirements.

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

7.3.3.1 Boundary Value Analysis

Boundary value analysis is a software testing technique in which tests are designed to include representatives of boundary values in a range. The idea comes

from the boundary. Given that we have a set of test vectors to test the system, a topology can be defined on that set.

7.3.3.2 Equivalence partitioning

Equivalence partitioning or equivalence class partitioning (ECP) is a software testing technique that divides the input data of a software unit into partitions of equivalent data from which test cases can be derived. In principle, test cases are designed to cover each partition at least once.

7.3.3.3 Comparison Testing

Comparison testing involves comparing the contents of databases files, folders, etc. A comparison testing can be carried out in two ways. Direct comparison testing: It is a testing of particular parts of each site against each other. Usually, multiple sites are compared side by side. Objective comparison testing: Execute the same test for each site separately. Usually, one site at a time is tested.

7.4 Test Strategy and Approach

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For the reason a template for software testing a set step into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works “outward” toward the integration of the entire computer base system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.

7.4.1 Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is “putting them together” interfacing. There may be the chances of data lost across on another’s sub functions, when combined may produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; goals data structure can present problems.

7.4.2 User Acceptance Testing

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developed.

CHAPTER 8

RESULT AND DISCUSSION

The success rate of data collection assuming that the nodes in each area are randomly and uniformly distributed. They aim to maximize the success rate of data collection by identifying the trust degree of nodes and selecting nodes with the highest degree of trust as aggregators. By doing so, they can minimize the probability of data being attacked and ensure high security performance of data collection. And also find the best optimal path to send the data packet to UAV.

8.1 PERFORMANCE ANALYSIS

To evaluate the performance of CRTC solution and optimization algorithms, we implement experiments using Python. We simulate a 100 nodes network deployment scenario, where a certain number of smart nodes are randomly distributed, as shown in Fig. 9(a). We set up different node distribution scenarios and repeatedly calculate UAV's flight distance to guarantee the effectiveness of proposed algorithm.

8.1.1 UAV PATH OPTIMIZATION

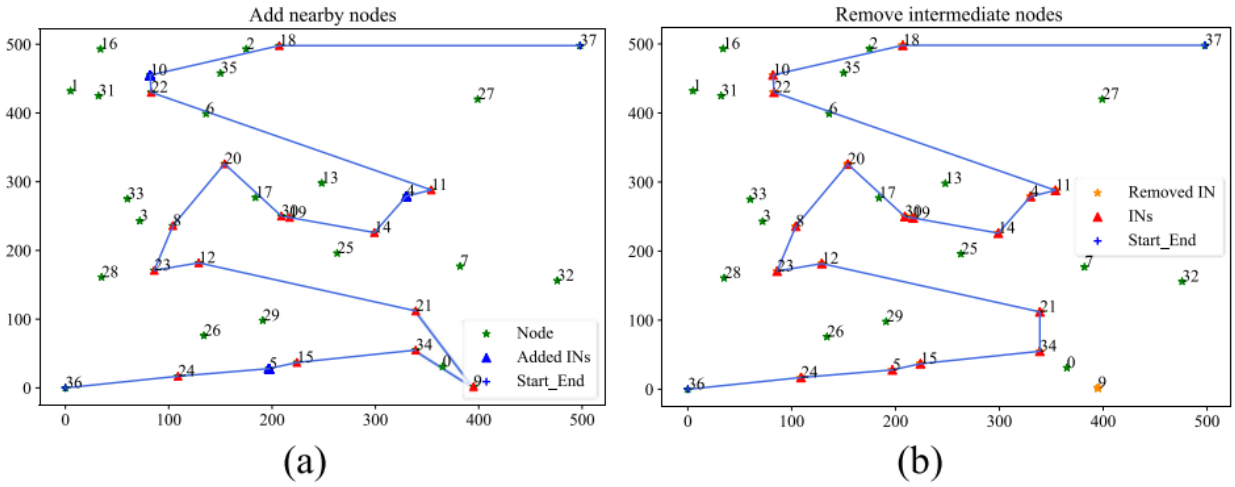
we obtain the optimal route before using the proposed optimization algorithm based on ant colony algorithm and 2-OPT technology [3]. The number of iterations is N = 100. It can be seen from Fig. 9(b) that some nodes, such as IN17 are close to the UAV but not used as INs. Therefore, we try to add these nodes as INs. Because they are close to the UAV, UAV does not need to spend extra energy, or just consume a little energy, it can increase the number of INs, thereby reducing the burden of INs and balancing network energy consumption.

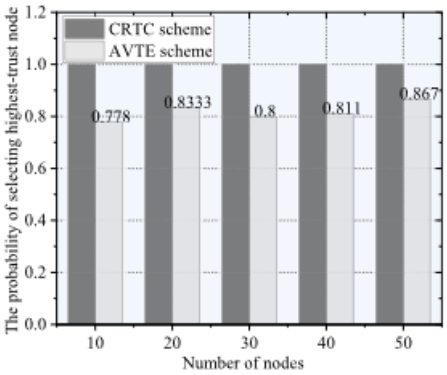
8.1.2 UAV Flight Distance

The flight distance of the UAV and the ratio of increase or decrease in flight length. In previous paper N = 100, the maximum distance is only increased by 4.25%, but it is reduced by 4.79% after removing IN. Therefore, our algorithm can still optimize the flight route while increasing nearby nodes.

We optimized the path distances. .Compared with the traditional ant colony algorithm, the proposed algorithm greatly reduces the flight distance by 64.40%. Therefore, our proposed algorithm can balance the energy consumption of IN while effectively reducing UAV trajectory distance. We find a best optimal path in 1.43s/it for 100 nodes in 200m x 200m area.

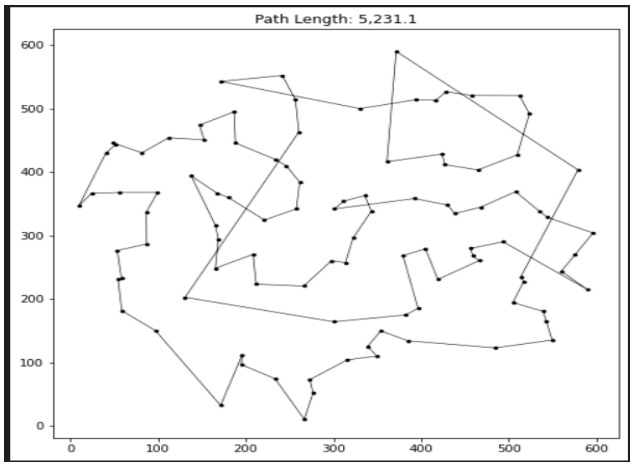
In previous paper, route path of n=100



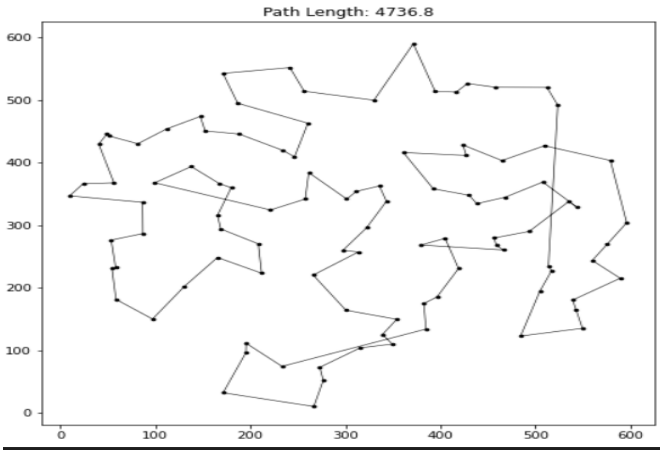


In current paper, the route path of n=100

Path before optimization



Path after optimization



RESULT

- 40% reduced the intermediate node119/300 [02:50<04:19, 1.43s/it]
- breaking at iteration: 119 with best path length: 4736.8152396678925

CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENT

9.1 CONCLUSION

Based on the information provided in the PDF, the proposed Cloud-Assisted Reliable Trust Computing Scheme for Data Collection in Internet of Things aims to address the challenge of ensuring secure and reliable data collection in IoT environments. The scheme uses a combination of trust evaluation, node selection, and UAV optimization techniques to maximize the success rate of data collection while minimizing the risk of attacks. The proposed scheme introduces a new method for obtaining reliable trust by comparing data obtained from source ISDs with data collected through other methods. This approach is innovative and has not been used in previous trust evaluations. While the proposed scheme shows promise, there are several areas for future work that could enhance its effectiveness. These include integrating blockchain technology, incorporating real-time feedback for dynamic trust evaluation, exploring multi-UAV coordination, and implementing privacy-preserving techniques.

FUTURE WORK

The study highlighted the potential for further advancements in UAV technology. Increasing the UAV's endurance and payload capacity would expand its capabilities and enable more extensive missions. Improvements in autonomous navigation and obstacle avoidance algorithms would enhance its safety and reliability. Additionally, ongoing research in miniaturized sensors, advanced imaging technologies, and data processing algorithms would contribute to enhanced data collection, analysis, and interpretation.

REFERENCE

[1] X. Zhang and Z. Ge, “Local parameter optimization of LSSVM for industrial soft sensing with big data and cloud implementation,” IEEE Trans. Ind. Informat., vol. 16, no. 5, pp. 2917–2928, May 2020, doi: 10.1109/TII.2019.2900479.

[2] J. Huang, L. Kong, G. Chen, M. Wu, X. Liu, and P. Zeng, “Towards secure industrial IoT: Blockchain system with credit-based consensus mechanism,” IEEE Trans. Ind. Informat., vol. 15, no. 6, pp. 3680–3689, Jun.2019,doi:10.1109/TII.2019.2903342.

[3] M. Shen, A. Liu, G. Huang, N. Xiong, and H. Lu, “ATTDC: An active and Trace-able trust data collection scheme for industrial security in smart cities,” IEEE Internet Things J., vol. 8, no. 8, pp. 6437–6453, Apr. 2021.

[4] S. Huang, A. Liu, S. Zhang, T. Wang, and N. Xiong, “BD-VTE: A novel baseline data based verifiable trust evaluation scheme for smart network systems,” IEEE Trans. Netw. Sci. Eng., to be published, doi:10.1109/TNSE.2020.3014455.

[5] P. Mishra, V. Varadharajan, E. S. Pilli, and U. Tupakula, “VMGuard: A VMI-Based security architecture for intrusion detection in cloud environment,” IEEE Trans. Cloud Comput., vol. 8, no. 3, pp. 957–971, Jul.–Sep.2020,doi:10.1109/TCC.2018.2829202.

[6] T. Wang, H. Luo, W. Jia, A. Liu, and M. Xie, “MTES: An intelligent trust evaluation scheme in sensor-cloud-enabled industrial Internet of Things,” IEEE Trans. Ind. Informat., vol. 16, no. 3, pp. 2054–2062, Mar. 2020, doi:10.1109/TII.2019.2930286.

[7] P. Mishra, E. S. Pilli, V. Varadharajan, and U. Tupakula, “Intrusion detection techniques in cloud environment: A survey.” J. Netw. Comput. Appl., vol.77,pp.18–47,2017.

[8] K. Huang, Q. Zhang, C. Zhou, N. Xiong, and Y. Qin, “An efficient

intrusion detection approach for visual sensor networks based on traffic pattern learning," IEEE Trans. Syst., Man, Cybern. Syst., vol. 47, no. 10, pp.2704–2713,Oct.2017.

[9] Y. Liu, A. Liu, X. Liu, and M. Ma, "A Trust-based active detection for Cyber-physical security in industrial environments," IEEE Trans. Ind. Informat.,vol.15,no.12,pp.6593–6603,Dec.2019.

[10] D. C. Mehetre, S. E. Roslin, and S. J. Wagh, "Detection and prevention of black hole and selective forwarding attack in clustered WSN with active trust," Cluster Comput., vol. 22, no. 1, pp. 1313–1328, 2019.

[11] Y. Liu, M. Ma, X. Liu, N. Xiong, A. Liu, and Y. Zhu, "Design and analysis of probing route to defense sink-hole attacks for Internet of Things security," IEEE Trans. Netw. Sci. Eng., vol. 7, no. 1, pp. 356–372, Jan.–Mar.2020.

[12] Z. Kazemi, A. Safavi, F. Naseri, L. Urbas, and P. Setoodeh, "A secure hybrid dynamic state estimation approach for power systems under false data injection attacks," IEEE Trans. Ind. Informat., vol. 16, no. 12, pp.7275–7286,Dec.2020.

[13] M. Dong, K. Ota, L. T. Yang, A. Liu, and M. Guo, "LSCD: A low storage clone detecting protocol for cyber-physical systems," IEEE Trans. Compute. AidD.,vol.35,no.5,pp.712–723,May2016.

[14] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for in-trusion detection," IEEE Commun. Surv. Tut., vol. 21, no. 1, pp. 686–728, Jan.–Mar.2018.

[15] X. Zhu, Y. Luo, A. Liu, M. Z. A. Bhuiyan, and S. Zhang, "Multi-agent deep reinforcement learning for vehicular computation offloading in IoT," IEEE Internet Things J., vol. 8, no. 12, pp. 9763–9773, Jun. 2021, doi: 10.1109/JIOT.2020.3040768.

[16] D. Gaushell and H. Darlington, "Supervisory control and data acquisition,"

Proc. IEEE, vol. 75, no. 12, pp. 1645–1658, Jan. 1988.
[17] D. Ebrahimi, S. Sharafeddine, P. H. Ho, and C. Assi, "UAV-aided projection-based compressive data gathering in wireless sensor networks." IEEE Internet Things J., vol. 6, no. 2, pp. 1893–1905, Apr. 2018.
[18] Y. Liu, M. Dong, K. Ota, and A. Liu, "ActiveTrust: Secure and trustable routing in wireless sensor networks," IEEE Trans. Inf. Forensics Secure, vol.11,no.9,pp.2013–2027,Sep.2016.
[19] W. Mo, T. Wang, S. Zhang, and J. Zhang, "An active and verifiable trust evaluation approach for edge computing," J. Cloud Comput., vol. 9, 2020.
[20] C. Huang, G. Huang, W. Liu, R. Wang, and M. Xie, "A parallel joint optimized relay selection protocol for wake-up radio enabled WSNs," Phys. Commun., vol. 47, Aug. 2021, Art. no. 101320.

APPENDIX

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from pathlib import Path

from tqdm import trange

plt.rcParams['figure.figsize']=[8,8]
```

Read point

```
def read_points(fl: str):
    "Reads points from text file"
    fl = Path(fl)
    points = []
    with fl.open() as f:
        line = f.readline()
        line = f.readline()
        while line:
            x,y = line.split()
            points.append((float(x), float(y)))
            line = f.readline()
    return np.array(points, dtype=np.float32)
```

Ants optimization

```
class AntOpt():
    def __init__(self,
                 points,
                 d_matrix = None,
                 dist='euclid', # distance metric
                 n_iter=300,    # Number of iterations
                 n_ants=10,     # Number of ants
                 alpha=2,       # pheromone importance
                 beta=3,        # local importance heuristic
                 rho=0.85,      # evaporation factor
                 Q=0.3,         # pheromone amplification factor
                 tau0=1e-4      # initial pheromone level
                ):
        self.n_iter = n_iter
        self.n_ants = n_ants
        self.alpha = alpha
        self.beta = beta
        self.rho = rho
        self.Q = Q
```

```
    self.tau0 = tau0
    self.points = points
    self.n_points = len(self.points) # number of nodes/cities
    self.cities = np.arange(self.n_points) # list of nodes/cities
    self.dist = dist
    if d_matrix is None:
        self.d_matrix = self.calc_distance_matrix(self.points)
    else:
        self.d_matrix = d_matrix
    # Check distance matrix is symmetric
    assert (self.d_matrix == self.d_matrix.transpose()).all()
    self.pheremons = self.tau0*np.ones_like(self.d_matrix)
    np.fill_diagonal(self.pheremons, 0)  #  no transition to the same node
    # set seed
    np.random.seed(0)
@staticmethod
def haversine_distance(lon1, lat1, lon2, lat2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
```

```
    Reference:
        https://stackoverflow.com/a/29546836/7657658
    """
    lon1, lat1, lon2, lat2 = map(np.radians, [lon1, lat1, lon2, lat2])
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = np.sin(
        dlat / 2.0)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(dlon / 2.0)**2
    c = 2 * np.arcsin(np.sqrt(a))
    km = 6371 * c
    return km
@staticmethod
def euclid_distance(p1, p2):
    "Calculate Euclidean distance between two points in 2d"
    assert p1.shape
    return np.sqrt((p1[0]-p2[0])**2 + (p1[1]-p2[1])**2)
def calc_distance(self, p1, p2, dist='euclid'):
    """
    Calculate distance between two points
    dist: distance metric [euclid or geo]
```

```
    """
    if dist == 'euclid':
        return self.euclid_distance(p1, p2)
    elif dist == 'geo':
        return self.haversine_distance(p1[0], p1[1], p2[0], p2[1])
    else:
        raise('Unknown distance metric, use euclid or geo')
def calc_distance_matrix(self, points: np.array):
    "Calculate distance matrix for array of points"
    n_points = len(points)
    d_matrix = np.zeros((len(points), len(points)), dtype=np.float32)
    for i in range(n_points):
        for j in range(i):
            d_matrix[i,j] = self.calc_distance(points[i,:], points[j, :], dist=self.dist)
    return d_matrix + d_matrix.transpose()  # symmetric
def path_length(self, path):
    tot_length = 0
    for i in range(len(path)-1):
        tot_length += self.d_matrix[path[i],path[i+1]]
    return tot_length
```

```
def _make_transition(self, ant_tour):
    "Make single ant transition"
    crnt = ant_tour[-1]
    options = [i for i in self.cities if i not in ant_tour]  # no repetition
    probs=np.array([self.pheremons[crnt,nxt]**
    self.alpha*(1/self.d_matrix[crnt,nxt])**self.beta for nxt in options])
    probs = probs/sum(probs)  # normalize
    next_city = np.random.choice(options, p=probs)
    ant_tour.append(next_city)
def run_ants(self):
    "Run ants optimization"
    # Initizlize last improvement iteration
    last_iter = 0
    # Initizlie optimal length
    optimal_length = np.inf
    # Keep track of path length improvement
    best_path_lengths = []
    for it in trange(self.n_iter):
        paths = []
        path_lengths = []
```

```
# release ants
for j in range(self.n_ants):
    # Place ant on random city
    ant_path = [np.random.choice(self.cities)
    # Make ant choose next node until it covered all nodes
    self._make_transition(ant_path)
    while len(ant_path) < self.n_points:
        self._make_transition(ant_path)
    # Return to starting node
    ant_path += [ant_path[0]
    # Calculate path length
    path_length = self.path_length(ant_path)
    paths.append(ant_path)
    path_lengths.append(path_length)
    # Check if new optimal
    if path_length < optimal_length:
        optimal_path = ant_path
        optimal_length = path_length
        last_iter = it
    best_path_lengths.append(optimal_length)
```

```
            # Break if no improvements for more than 50 iterations
            if (it - last_iter) > 50:
                print(f'breaking at iteration: {it} with best path length: {optimal_length}')
                break
            # Evaporate pheromons
            self.pheremons = self.rho*self.pheremons
            # Update pheremons based on path lengths
            for path, length in zip(paths, path_lengths):
                for i in range(self.n_points - 1):
                    self.pheremons[path[i],path[i+1]] += self.Q/length
            # Elitist ant
            for k in range(self.n_points - 1):
                self.pheremons[optimal_path[k],optimal_path[k+1]] += self.Q/optimal_length
        return optimal_path
    def greedy(self):
        "Generate path by moving to closest node to current node"
        start = np.random.choice(self.cities)
        print(f"start: {start}")
        path = [start]
```

```
        while len(path) < len(self.cities):
            options = np.argsort(self.d_matrix[start,:])  # find nearest node
            nxt = [op for op in options if op not in path][0]
            start = nxt
            path.append(nxt)
        # return home
        path += [path[0]]
         return path
       def plot_cities(self):
        "Plot the nodes"
        plt.scatter(self.points[:, 0], self.points[:, 1], s=7, color='k')
        plt.axis('square');
        def plot_path(self, path):
        "Plot a path"
        self.plot_cities()
        plt.plot(self.points[path,0], self.points[path,1], color='k', linewidth=0.6)
        plt.title(f'Path Length: {self.path_length(path):,.1f}')
    def __repr__(self):
        return f"Optimizing with {self.n_points} cities, n_iter={self.n_iter}, n_ants={self.n_ants}, alpha={self.alpha}, beta={self.beta}, rho={self.rho}, Q={self.Q}"
```

Optimizing with 10 cities, n_iter=300, n_ants=10, alpha=2, beta=3, rho=0.85, Q=0.3

```
ants = AntOpt(points10)

ants

# greedy solution

greedy_path = ants.greedy()

ants.plot_path(greedy_path)

plt.savefig('images/10nodes_tsp_greedy.png',bbox_inches='tight',
transparent=True)

ants = AntOpt(points10)

best_path = ants.run_ants()

ants.plot_path(best_path)

plt.savefig('images/10nodes_tsp.png', bbox_inches='tight', transparent=True)
```

Optimizing with 100 cities, n_iter=300, n_ants=15, alpha=2, beta=3, rho=0.85, Q=0.3

```
 ants = AntOpt(points100, n_ants=15)

ants

# greedy solution

greedy_path = ants.greedy()

ants.plot_path(greedy_path)
```

```
ants = AntOpt(points100, n_ants=20)

best_path = ants.run_ants()

ants.plot_path(best_path)

plt.savefig('images/100nodes_tsp.png', bbox_inches='tight', transparent=True)
```

THANK YOU!!