RECTILINEAR MINIMAL STEINER TREE PROBLEM: SOLVING THROUGH PIGEON INSPIRED OPTIMIZATION

PROJECT REPORT
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE
DEGREE OF BACHELOR OF TECHNOLOGY
IN
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BY

NAME	Registration No:	Roll No:
CHANDRODOY BANERJEE	161420110018	14200116052
SAPTARSHI BISWAS	161420110043	14200116027
SOURADIP SANTRA	161420110050	14200116020
SUVRAJIT DEB	161420110058	14200116012

Under The Supervision
of
Subhrapratim Nath
Head of Department
Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MEGHNAD SAHA INSTITUTE OF TECHNOLOGY
Techno Complex, Madurdaha, Beside NRI Complex, post-Uchhepota, Kolkata-700150

Affiliated by
MAULANA ABUL KALAM AZAD
UNIVERSITY OF TECHNOLOGY,
WEST BENGAL

JUNE 2020

CANDIDATE'S DECLARATION

I hereby declare that the work which is being presented in the project entitled "**Rectilinear Minimal Steiner Tree Problem: Solving Through Pigeon Inspired Optimization"** in fulfillment of requirements for the award of degree of B.Tech. in CSE, submitted in the Department of Computer Science & Engineering at **MEGHNAD SAHA INSTITUTE OF TECHNOLOGY** under **MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, KOLKATA** is an authentic record of our own work carried out during Session 2019-2020 under the supervision of **Subhrapratim Nath, Head of Department, CSE, MSIT.** The matter presented in this project has not been submitted by us in any other University / Institute for any award.

25.06.2020

Chandrodoy Banerjee
Saptarshi Biswas
Sowradip Santra
Suvrajit Deb

Signature of the Students
With Date

Meghnad Saha Institute of Technology

Nazirabad, P.O. :Utchepota, Via Sonarpur, Kolkata 700 150

CERTIFICATE

This is to certify that the Project entitled **Rectilinear Minimal Steiner Tree Problem: Solving Through Pigeon Inspired Optimization** is being submitted by **Chandrodoy Banerjee, Saptarshi Biswas, Souradip Santra and Suvrajit Deb** in partial fulfillment of the requirement for the award of the degree of B.Tech.in Computer Science and Engineering to the Department of Computer Science and Engineering, Meghnad Saha Institute of Technology, Kolkata, is a record of original work carried out by him under my guidance and supervision from July 2019 to June 2020.

The results presented in this thesis have been verified and are found to be satisfactory. The results embodied in this thesis have not been submitted to any other University for the award of any other degree or diploma.

Subhrapratim Nath
Head of Department
Computer Science and Engineering
Meghnad Saha Institute of Technology
Kolkata-700150

Date: **June 25, 2020**

Place: **Kolkata**

Meghnad Saha Institute of Technology

Nazirabad, P.O. :Utchepota, Via Sonarpur, Kolkata 700 150

CERTIFICATE OF APPROVAL

The foregoing project entitled **Rectilinear Minimal Steiner Tree Problem: Solving Through Pigeon Inspired Optimization** is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the degree for which it has been submitted. It is to be understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

Project Guide _____

Head of the Department _____

Date: **June 25, 2020**

Place: **Kolkata**

Acknowledgement

Life as a researcher is a zig-zag way and has many ups and downs, but I overcome them because of the support and faith of many individuals.

First and foremost I would like to express my sincere gratitude to **Subhrapratim Nath** for providing me an opportunity to work with him/her. Your creativity encourages me, your energy and dedication towards work motivates me. Your perfection always inspired me to do things perfectly. I feel motivated and encouraged every time I attend your meeting. Without your encouragement, inspiration and guidance this thesis/dissertation would not have materialized. Thank you so much for not only guiding me in this thesis / dissertation but for every knowledge you give me. This thesis / dissertation is dedicated to your creativity and enthusiasm of doing work.

I wish to thank many people for their direct or indirect help in achieving this goal. Special thanks to my parents for their support and encouragement throughout my study, their faith in me ignited that spark to do anything for their happiness.

Last but not least I would like to thank all my friends for being with me at each step when I need their support. This thesis will never be successful without your support and love.

25.06.2020

(Signature of the Students)

Abstract

With the rapid advances in Very large scale integration (VLSI) technology, it is important to optimize the wire length of the circuits so that the performance of the circuits can be enhanced. Routing is one of the most important stages in the physical layer design of VLSI circuits. The VLSI Global Routing problem aims to minimize the total interconnection length by mapping it to the Rectilinear Steiner Minimal Tree (RSMT) Problem which finds a set of additional points, called Steiner points, such that the length of a rectilinear minimum spanning tree constructed with n terminal points is minimized. The terminal points are nothing but various active and passive components in the VLSI circuits. A heuristic approach is used in optimizing using the Pigeon Inspired Optimization (PIO), which has the potential to solve the RSMT problem. Complexity analysis and simulation results were compared with the conventional Particle Swarm Optimization (PSO) algorithm. A comparative study has been done on both the versions and the Pigeon Inspired Optimization (PIO) has been found to be better in optimizing the wire lengths.

Keywords:Metaheuristics, Rectilinear Steiner Tree Optimization, Pigeon Inspired Optimization, Greedy Algorithm

TABLE OF CONTENTS

Page No.

Abstract

CHAPTER 1. INTRODUCTION

During the last few decades, the world has witnessed a surge in the use of electronic goods. However, this surfeit has introduced a new challenge related to the energy and cost optimisation of the electronic devices. As a result, researchers primarily tried to solve this problem through

algorithmic approaches. With the rapid advances in VLSI technology, it is important to optimize the wire length of the circuits. This proposed work focuses on solving the Rectilinear Steiner Minimal Tree (RSMT) problem with the help of a metaheuristic algorithm, namely, the Pigeon Inspired Optimization (PIO) algorithm. The performance is compared with Particle Swarm Optimization based VLSI routing. The results obtained show the proposed methodology has a good potential in VLSI routing and can be further extended in future.

1.1 Very Large Scale Integration:

Very-Large-Scale Integration (VLSI) is the process of creating an integrated circuit (IC) by combining thousands of transistors into a single chip. VLSI began in the 1970s when complex semiconductor and communication technologies were being developed. The microprocessor is a VLSI device. Before the introduction of VLSI technology most ICs had a limited set of functions they could perform. An electronic circuit might consist of a CPU , ROM , RAM and other glue logic . VLSI lets IC designers add all of these into one chip Structured VLSI design is a modular methodology originated by Carver Mead and Lynn Conway for saving microchip area by minimizing the interconnect fabrics area. This is obtained by repetitive arrangement of rectangular macro blocks which can be interconnected using wiring by abutment . An example is partitioning the layout of an adder into a row of equal bit slices cells. In complex designs this structuring may be achieved by hierarchical nesting. Structured VLSI design had been popular in the early 1980s, but lost its popularity later because of the advent of placement and routing tools wasting a lot of area by routing , which is tolerated because of the progress of *Moore's Law. When introducing the hardware description language KARL in the mid' 1970s, Reiner Hartenstein coined the term "structured VLSI design" (originally as "structured LSI design"), echoing Edsger Dijkstra's structured programming approach by procedure nesting to avoid chaotic spaghetti-structured programs.

**Moore's Law: -In 1965, Gordon Moore (Fairchild) stated that the number of transistors on an IC would double every year. 10 years later, he revised his statement, asserting that they double every 18 months. Since then, this "rule" has been famously known as Moore's Law.*

1.2 Physical Design:

In integrated circuit design , physical design is a step in the standard design cycle which follows after the circuit design . At this step, circuit representations of the components (devices and interconnects) of the design are converted into geometric representations of shapes which, when manufactured in the corresponding layers of materials, will ensure the required functioning of the components. This geometric representation is called integrated circuit layout . This step is usually split into several sub-steps, which include both design and verification and validation of the layout.

Modern day Integrated Circuit (IC) design is split up into Front-end design using HDLs, Verification , and Back-end Design or Physical Design . The next step after Physical Design is the Manufacturing process or Fabrication Process that is done in the Wafer Fabrication Houses. Fab-houses fabricate designs onto silicon dies which are then packaged into ICs.

1.2.1 Floor-planning:

Physical design begins with a floor-plan. The floor-plan estimates the area of major units in the chip and defines their relative placements. The floor-plan is essential to determine whether a proposed design will fit in the chip area budgeted and to estimate wiring lengths and wiring congestion, so an initial floor-plan should be prepared as soon as the logic is loosely defined. Floor-planning is an essential design step for hierarchical building module design methodology. Floor-planning provides early feedback that evaluates architectural decisions; estimates chip area, estimates delay and congestion caused by wiring. For many years, floor-planning is a critical step, as it sets up the groundwork for a layout. However, it is computational quite hard. The process of determining block shapes and positions with area minimization objective and aspect ratio requirement is referred to as floor-planning.

Consider a large design like a microprocessor. The abstract level behavioral description of the processor is written using an RTL program . Large designs (e.g. microprocessor in our case) are usually synthesized into small modules. These modules are the basic building blocks of a microprocessor e.g. memory unit, adder/subtractor (ALU) unit, multiplexer unit, etc. Consider the following diagram

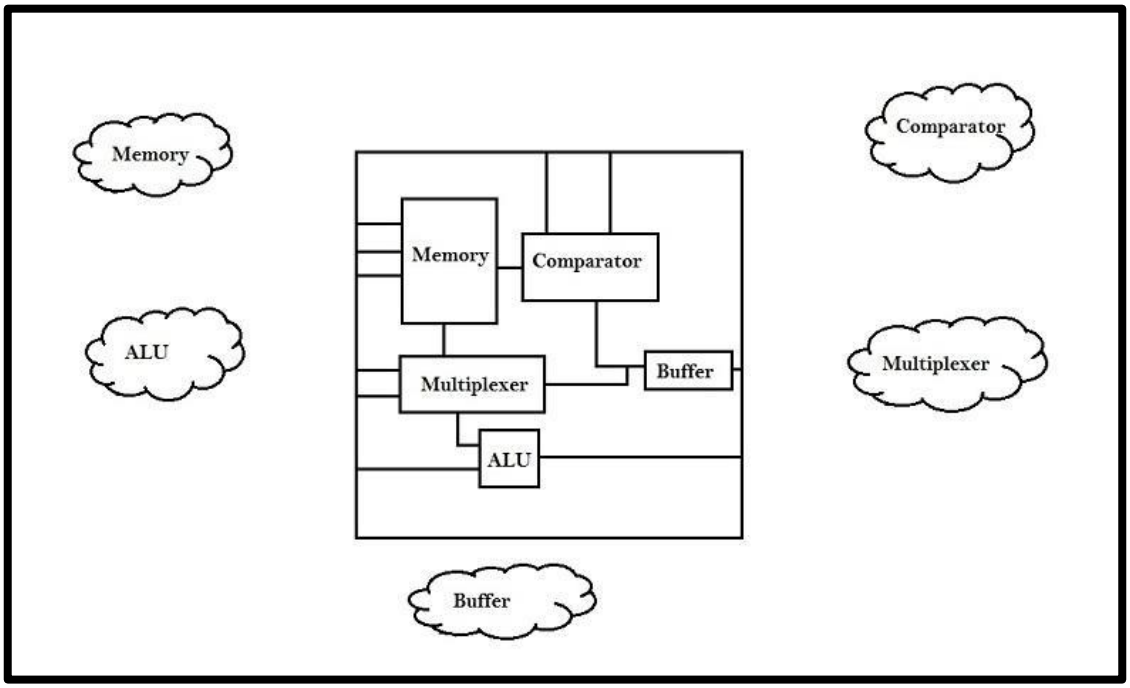

Fig 1: A Typical Microprocessor Unit

Floor-planning makes sure of 3 things:
1) Every module has been assigned an appropriate area and aspect ratio.
2) Every pin of the module has connection with other modules or periphery of the chip.
3) Modules are arranged in a way such that it consumes less area on a chip.

1.2.2 Partitioning:

Partitioning is a process of dividing the chip into small blocks. This is done mainly to separate different functional blocks and also to make placement and routing easier. Partitioning can be done in the RTL design phase when the design engineer partitions the entire design into sub-blocks and then proceeds to design each module. These modules are linked together in the main module called the TOP LEVEL module. This kind of partitioning is commonly referred to as Logical Partitioning.

1.2.3 Placement:

Before the start of placement optimization all Wire Load Models (WLM) are removed. Placement uses RC values from Virtual Route (VR) to calculate timing. VR is the shortest Manhattan distance between two pins. VR RCs are more accurate than WLM RCs.

Placement is performed in four optimization phases:

1. Pre-placement optimization
2. In placement optimization
3. Post Placement Optimization (PPO) before clock tree synthesis (CTS)
4. PPO after CTS.

- Pre-placement Optimization optimizes the net-list before placement, HFNs are collapsed. It can also downsize the cells.

- In-placement optimization re-optimizes the logic based on VR. This can perform cell sizing, cell moving, cell bypassing, net splitting, gate duplication, buffer insertion, area recovery. Optimization performs iteration of setup fixing, incremental timing and congestion driven placement.

- Post placement optimization before CTS performs net-list optimization with ideal clocks. It can fix setup, hold, max trans/cap violations. It can do placement optimization based on global routing. It re does HFN synthesis.

- Post placement optimization after CTS optimizes timing with propagated clock. It tries to preserve clock skew.

1.2.4 Clock tree synthesis :

The goal of clock tree synthesis (CTS) is to minimize skew and insertion delay. Clock is not propagated before CTS. After CTS holds, slack should improve. Clock tree begins at .sdc defined clock source and ends at stop pins of flop. The clock tree optimization (CTO) clock can be shielded so that noise is not coupled to other signals. But shielding increases area by 12 to 15%. Since the clock signal is global in nature the same metal layer used for power routing is used for clock also.

1.2.5 Physical Verification

Physical verification is a process whereby an IC layout design is checked via EDA software tools to see if it meets certain criteria. It checks the correctness of the generated layout design. This includes verifying that the layout

1. Complies with all technology requirements – Design Rule Checking (DRC)
2. Is consistent with the original net-list – Layout vs. Schematic (LVS)
3. Has no antenna effects – Antenna Rule Checking
4. This also includes density verification at the full chip level...Cleaning density is a very critical step in the lower technology nodes
5. Complies with all electrical requirements – Electrical Rule Checking (ERC).

1.3 VLSI Physical Design Optimization:

1.3.1 Types of Constraint:

- Technology constraints enable fabrication for a specific technology node and are derived from technology restrictions. Examples include minimum layout widths and spacing values between layout shapes.
- Electrical constraints ensure the desired electrical behavior of the design. Examples include meeting maximum timing constraints for signal delay and staying below maximum coupling capacitances.
- Geometry (design methodology) constraints are introduced to reduce the overall complexity of the design process. Examples include the use of preferred wiring directions during routing, and the placement of standard cells in rows.

1.3.2 Runtime Complexity:

- Runtime complexity: the time required by the algorithm to complete as a function of some natural measure of the problem size, allows comparing the scalability of various algorithms
- Complexity is represented in an asymptotic sense, with respect to the input size n, using big-Oh notation or O(…)
- Runtime $t(n)$ is order $f(n)$, written as $t(n) = O(f(n))$

- Example: $t(n) = 7n! + n^2 + 100$, then $t(n) = O(n!)$
  because $n!$ is the fastest growing term as n□ .
- A number of physical design problems have best-known algorithm complexities that grow exponentially with n, e.g., $O(n!)$, $O(n^n)$, and $O(2^n)$.
- Many of these problems are NP-hard (NP: non-deterministic polynomial time)
- No known algorithms can ensure, in a time-efficient manner, globally optimal solution
- *Heuristic algorithms* are used to find near-optimal solutions

1.4. Purpose

The Rectilinear Steiner Tree problem arises in the domain of physical design in the field of electronic design automation. In VLSI circuits, wire routing is carried out by wires that are often constrained by design rules to run only in vertical and horizontal directions. So, the rectilinear steiner tree problem can be used to emulate the routing of nets with more than two terminals. An additional objective arises in the case of rectilinear steiner tree creation, which curtails the error ratio of the rectilinear steiner tree and the minimal spanning tree.

The purpose of the project is to optimize the generation of the rectilinear steiner minimal tree, which entails the generation of steiner points, using a meta-heuristic algorithm. This objective thus produces a minimal spanning tree, involving the generated steiner points. It will result in a reduced interconnection of wires, which run in vertical and horizontal directions and are used to connect net terminals, thus reducing energy loss, and also minimizing wire interconnect cost.

1.5 Domain Definition

The domain of Electronic Design Automation entails many optimization problems, of which wire routing is a notable one. In VLSI circuits, the wires connect node terminals, by running in a vertical-horizontal fashion. However, the cost of the connections, if carried out on a minimal spanning tree, is not minimal. Introduction of steiner points onto the terminals layout and using them as connection points can significantly reduce the total cost of wire connectivity. But the generation of steiner minimal trees is one of Karp's original 21 NP-complete problems. Meta-heuristic algorithms are generally used to provide near-optimal solutions to NP-complete problems. Thus, a metaheuristic algorithm based upon the behaviour of pigeons has been developed to improve the approximation ratio of the steiner tree problem.

1.6 Motivation

Rectilinear Steiner Minimal tree is one of Karp's original 21 NP-complete problems. The motivation of the project is to reduce the wire interconnect cost, by introducing optimal steiner points. The meta-heuristic algorithm employed strives to introduce steiner points, which will not only reduce the wire cost, but also produce a much lesser approximation ratio.

Reducing the wire cost bears practical advantages. When wire-cost is reduced by introducing steiner points, these points can be used in real life as virtual nodes for interconnecting given terminals. It reduces the total wire-cost. Also, the amount of energy loss is decreased due to reduced interconnect. Thus, less power is used for the VLSI circuit which results from reduced wire connections. It also reduces the cost of design of the VLSI circuit.

CHAPTER 2. PRELIMINARIES

2.1 Routing

Routing refers to sending documents from one person to another for review, approval or reference. Routing paper documents through traditional inter-organizational mail invites delays and missed communications. Electronic routing is to locate a set of wires in the routing space that connects all the nets in the net list. The capacities of channels, width of wire and wire crossing often needs to be taken into consideration.

Steps of Routing in physical design of VLSI:

- Apply after placement.
- Input
  - Net-list
  - Timing budget for, typically, critical nets
  - Location of blocks and location of pins.
- Output
  - Geometric layout of all nets
- Objective
  - Minimizing the wire length, number of *vias*, or just completing all connections without increasing the net area.
  - Each net meets its timing budget.

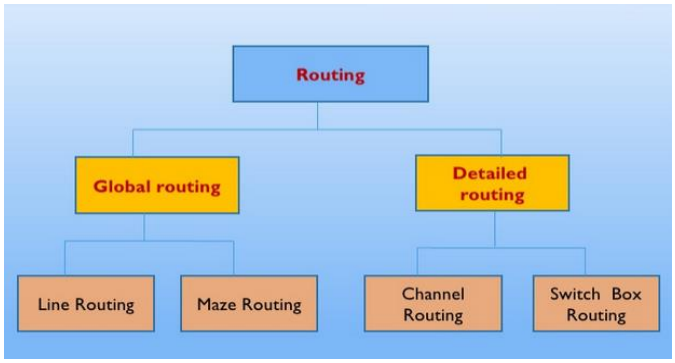

Fig 2: Routing Classification

2.1.1. Global Routing

Generate a 'loose' net for each net. Assign a list of routing regions to each net without specifying the actual layout of wires. The details of global routing differ slightly between cell-based ASICs, gate arrays, and FPGAs, but the principles are the same in each case. A global router does not make any connections, it just plans them. We typically global route the whole chip (or large pieces if it is a large chip) before detail routing the whole chip (or the pieces). There are two types of areas to the global route: inside the flexible blocks and between blocks (the Viterbi decoder, although a cell-based ASIC, only involved the global routing of one large flexible block).

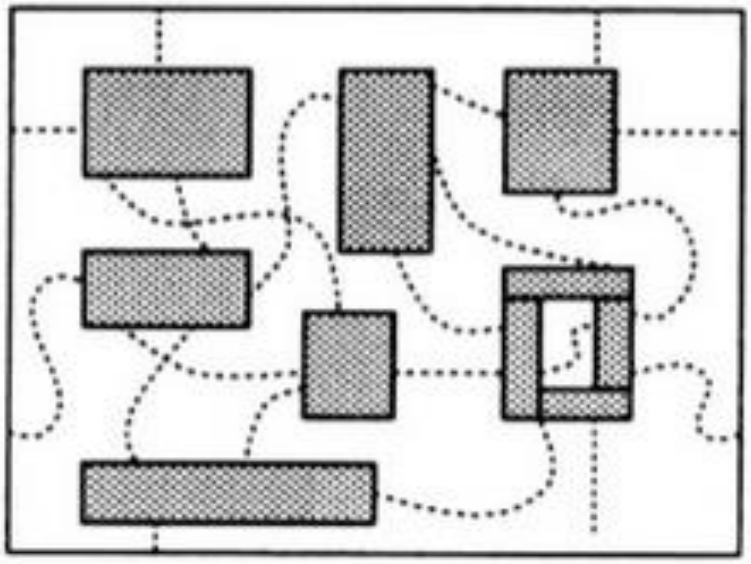

Fig 3: Global Routing

The input to the global router is a floor-plan that includes the locations of all the fixed and flexible blocks; the placement information for

flexible blocks; and the locations of all the logic cells. The goal of global routing is to provide complete instructions to the detailed router on where to route every net. The objectives of global routing are one or more of the following:

- Minimize the total interconnect length.
- Maximize the probability that the detailed router can complete the routing.
- Minimize the critical path delay.

In both floor-planning and placement, with minimum interconnect length as an objective, it is necessary to find the shortest total path length connecting a set of terminals. This path is the MRST, which is hard to find. The alternative, for both floor-planning and placement, is to use simple approximations to the length of the MRST (usually the half-perimeter measure). Floor-planning and placement both assume that interconnect may be put anywhere on a rectangular grid, since at this point nets have not been assigned to the channels, but the global router must use the wiring channels and find the actual path. Often the global router needs to find a path that minimizes the delay between two terminals—this is not necessarily the same as finding the shortest total path length for a set of terminals.

2.1.2. Detailed Routing

Find the actual geometry layout of each net within the assigned routing regions.

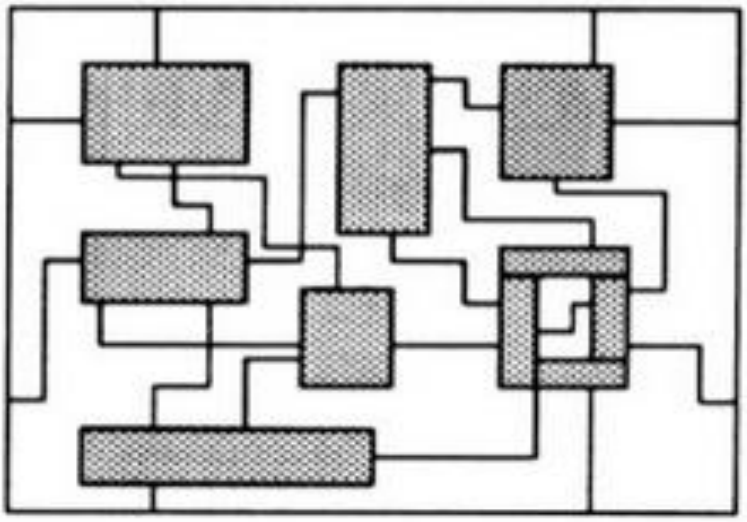

Fig 4: Detailed Routing

There are three types of detailed routing:

- Channel Routing

- 2-D Switchbox Routing
- 3-D Switchbox Routing

After global and detailed routing, the information of the net is extracted and delays can be analyzed.

2.2 **Graph Theory**

Graph theory is the study of *graphs*, which are mathematical structures used to model pairwise relations between objects. A "graph" in this context is made up of "vertices" or "nodes" and lines called *edges* that connect them. A graph may be *undirected*, meaning that there is no distinction between the two vertices associated with each edge, or its edges may be *directed* from one vertex to another.In computer science, graphs are used to represent networks of communication, data organization, computational devices, the flow of computation, etc. For instance, the link structure of a website can be represented by a directed graph, in which the vertices represent web pages and directed edges represent links from one page to another. A similar approach can be taken to problems in travel, biology, computer chip design, and many other fields. The development of algorithms to handle graphs is therefore of major interest in computer science.

2.2.1. Minimal Spanning Tree

Given a connected, undirected graph, a spanning tree of that graph is a subgraph that is a tree and connects all the vertices together. A single graph can have many different spanning trees. We can also assign a *weight* to each edge, which is a number representing how unfavorable it is, and use this to assign a weight to a spanning tree by computing the sum of the weights of the edges in that spanning tree. A minimum spanning tree (MST) or minimum weight spanning tree is then a spanning tree with weight less than or equal to the weight of every other spanning tree. More generally, any undirected graph (not necessarily connected) has a minimum spanning forest, which is a union of minimum spanning trees for its connected components.

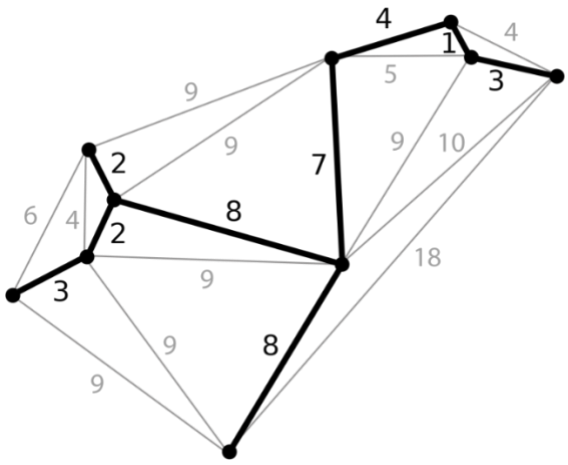

Fig 4: Minimum Spanning Tree of the given graph (MST in black bold line)

2.2.2. Rectilinear Minimal Spanning Tree

Given a set of points in a plane, a Rectilinear Minimum Spanning Tree (RMST) is a set of edges which connects all of the points with minimum rectilinear distance. Rectilinear distance (also called "Manhattan Distance") is the sum of the horizontal and vertical distance between two points. It is used instead of Euclidean distance in VLSI CAD applications because VLSI processes support only horizontal and vertical wires.

The problem commonly arises in physical design of electronic circuits. In modern high-density integrated circuits wire routing is performed by wires which consist of segments running horizontally in one layer of metal and vertically in another metal layer. As a result, the wire length between two points is naturally measured with rectilinear distance. Although the routing of a whole net with multiple nodes is better represented by the rectilinear Steiner tree, the RMST provides a reasonable approximation and wire length estimate.

- The RMST can be used as an estimate of the length of a net that connects together multiple terminals
- The RMST Problem is a special case of the Minimum Spanning Tree problem

***Manhattan Distance:** The distance between two points in a grid based on a strictly horizontal and/or vertical path (that is, along the grid lines), as opposed to the diagonal or "as the crow flies" distance. The Manhattan distance is the simple sum of the horizontal and vertical components, whereas the diagonal distance might be computed by applying the Pythagorean Theorem.*

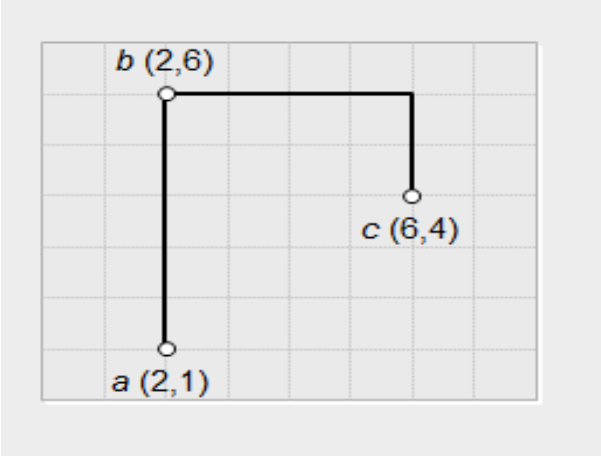

Fig 5: Rectilinear Minimum Spanning Tree

2.2.3. Rectilinear Steiner Tree

The routing of a whole net with multiple nodes is better represented by the Rectilinear Steiner tree, the RMST provides a reasonable approximation and wire length estimate. To know about Rectilinear Steiner Tree, let us discuss Steiner points first.

Steiner Points: A point that is not part of the input set of points, for instance, a point computed to construct a *Steiner tree*. It is a point with a particular geometric relation to a triangle. In the VLSI routing, to reduce the computational costing Steiner points i.e. the points besides the given nodes of the circuit, are introduced. Using these points the computational costing reduces to reach a particular node from a given source. It uses Rectilinear Steiner trees to reduce the costing.

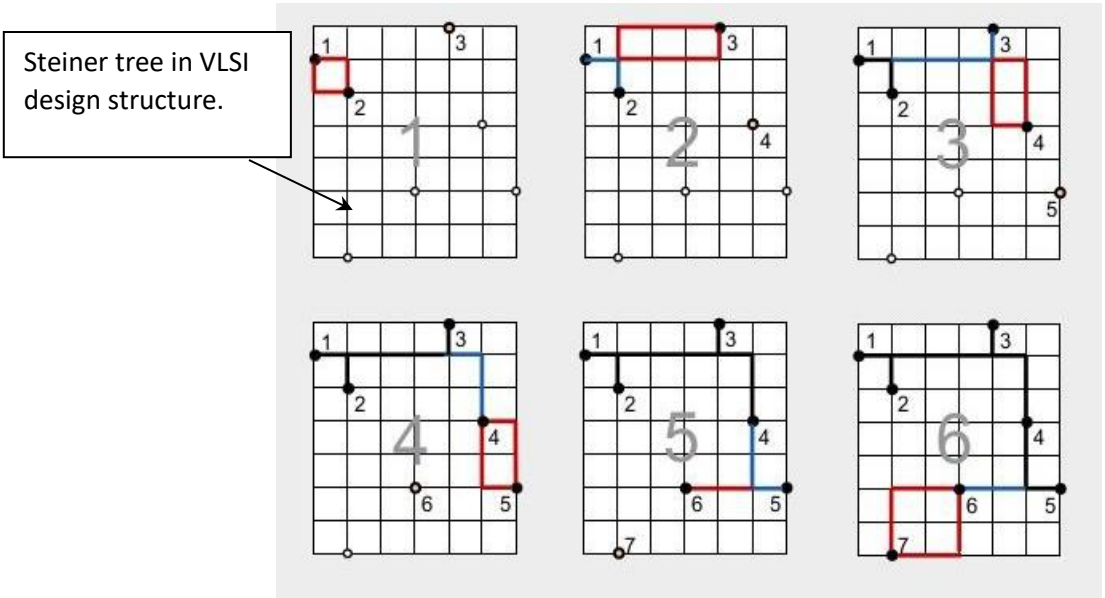

Fig 6: Steiner tree in VLSI design structure

Rectilinear Steiner tree ,minimum rectilinear Steiner tree problem (MRST), or rectilinear Steiner minimum tree problem (RSMT) is a variant of the geometric Steiner tree problem in the plane, in which the Euclidean distance is replaced with the rectilinear distance. Given a set of terminal points in a plane, a Minimum Steiner Rectilinear Tree (MSRT) is a set of edges which connects all the terminals along with a set of added "Steiner points" such that the rectilinear distance is minimized. F. Hwang proved that the length of a Steiner tree can be up to 3/2 times shorter than a Rectilinear Minimum Spanning Tree on the same set of terminals.

M. Hanan proved that a Minimum RST can be constructed using points on a grid defined by the x and y coordinates of the terminal points (now known as the Hanan Grid).

The Minimum Steiner Rectilinear tree problem is NP-Hard.

The Steiner tree problem is essentially a special case of multidimensional routing that can be defined as: for any given set T of n terminal points, the objective is to find another set S of additional points called Steiner points such that the length of a minimum rectilinear spanning tree or RSMT is minimized. As already mentioned our ultimate goal is the reduction of interconnect length, the RSMT problem becomes our fundamental problem also. The RSMT problem has been proved to be NP-complete.

***Hanan Grid:** In geometry, the Hanan grid H(S) of a finite set S of points in the plane is obtained by constructing vertical and horizontal lines through each point in S.*

The main motivation for studying the Hanan grid stems from the fact that it is known to contain a rectilinear Steiner tree for *S*. It is named after Maurice Hanan, who was first to investigate the rectilinear Steiner minimum tree and introduced this graph

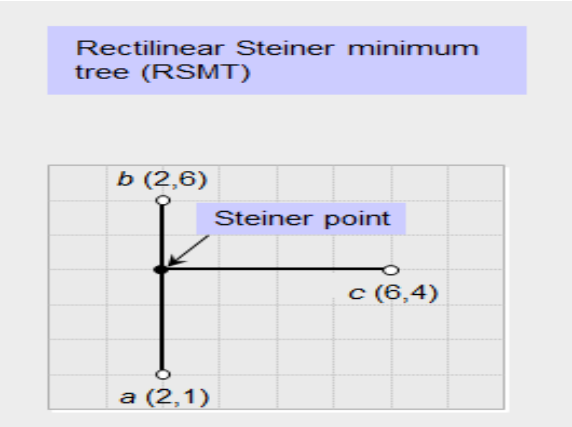

Fig 7: Rectilinear Steiner Minimum Tree (RMST)

2.3 Heuristic Algorithm

According to Schuster (1974):The heuristic approach to problem solving consists of applying human intelligence, experience, common sense and certain rules of thumb (or heuristics) to develop an acceptable, but not necessarily an optimum, solution to a problem. Of course, determining what constitutes an acceptable solution is part of the task of deciding which approach to use; but broadly defined, an acceptable solution is one that is both reasonably good (close to optimum) and derived within reasonable effort, time, and cost constraints. Often the effort (manpower, computer, and other resources) required, the time limits on when the solution is needed, and the cost to compile, process, and analyze all the data required for deterministic or other complicated procedures preclude their usefulness or favor the faster, simpler heuristic approach. Thus, the heuristic approach is generally used when deterministic techniques or are not available, economical, or practical.

2.3.1. Deterministic

All decisions made by the algorithm are repeatable, i.e., not random. One example of a deterministic heuristic is Dijkstra's shortest path algorithm.

- Stochastic: Some decisions made by the algorithm are made randomly, e.g., using a pseudo-random number generator. Thus, two independent runs of the algorithm will produce two different solutions with high probability. One example of a stochastic algorithm is simulated annealing.
- In terms of structure, a heuristic algorithm can be
  - Constructive: The heuristic starts with an initial, incomplete (partial) solution and adds components until a complete solution is obtained.
  - Iterative: The heuristic starts with a complete solution and repeatedly improves the current solution until a preset termination criterion is reached.

2.3.2. Objective

The objective of a heuristic is to produce a solution in a reasonable time frame that is good enough for solving the problem at hand. This solution may not be the best of all the actual solutions to this problem, or it may simply approximate the exact solution. But it is still valuable because finding it does not require a prohibitively long time.

Heuristics may produce results by themselves, or they may be used in conjunction with optimization algorithms to improve their efficiency (e.g., they may be used to generate good seed values).

Results about NP-hardness in theoretical computer science make heuristics the only viable option for a variety of complex optimization problems that need to be routinely solved in real-world applications.

2.3.3. Heuristic Routing

Heuristic routing is a system used to describe how data is delivered when problems in a network topology arise. Heuristic routing is achieved using specific algorithms to determine the best, although not always optimal, path to a destination. When an interruption in a network topology occurs, the software running on the networking electronics calculates another route to the desired destination via an alternate available path.

Heuristic routing*:* is also used for vehicular traffic using the highway and transportation network of the world, but that is beyond the scope of this article.

Routing in which data, such as time delay, extracted from incoming messages, during specified periods and over different routes, are used to determine the optimum routing for transmitting data back to the sources.

It allows a measure of route optimization based on recent empirical knowledge of the state of the network.

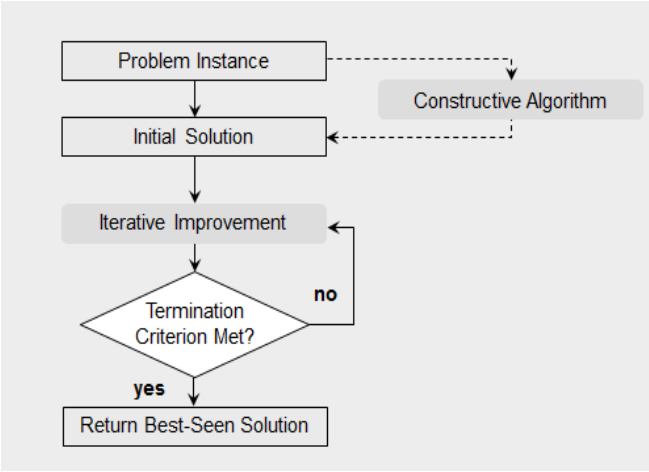

Fig 8: Flow chart of Heuristic routing algorithm

2.4 External Application Tool Requirement

2.4.1. Hardware Tools

All the algorithms were executed on a system running on a 64-bit Ubuntu 18.04.3 LTS as the OS. The device consisted of DDR3 RAM of 8GB. The processor used by the system was a quad-core AMD A6-7310 APU with a processing speed of 2 - 2.4 GHz.

2.4.2. Software Tools

All the algorithms were coded in Python 3.6 and its scientific library,

namely, Numpy. Visualization was done using Python plotting library, namely, Matplotlib. The simulation setup uses predefined datasets generated within the search space for 10, 20, 30, 40 and 50 terminal points respectively. The individual dataset includes the X and Y coordinates of the respective terminal points represented as a coordinate list within a Numpy file (.npy). All the algorithms were executed on the predefined dataset. It needs to be noted that no simulation softwares was used in this work. The algorithm was executed in python terminal and the results were collected and visualized in the standard runtime environment.

CHAPTER 3. LITERATURE REVIEWS

3.1 Steiner Tree Problem

On Steiner's Problem with Rectilinear Distance

- Author Name: M. Hanan
- Year of Publication: 1966
- Work Summary:

  This work demonstrates the use of Steiner Tree for VLSI global routing optimization. Steiner points are explicit points that can be introduced in a routing space to minimize the routing cost. The minimal routing cost in an electrical circuit can be calculated by tracing the minimal spanning tree connecting the terminal points. However, since the channel overlapping causes electrical signal interference which further causes to find out a unique path between two terminal points without mingling with other routing paths related to other terminal point pairs. This causes the routing cost to increase. In this context, the Steiner points were introduced which could act as switching elements which facilitated multipath routing through a single grid point. It elucidates the optimization objective based on the rectilinear search space. The work considers the rectilinear distance $d(p_1, p_2)$ between two points P_1 , P_2 is $|X_1 - X_2| + |Y_1 - Y_2|$, where the (x_i, y_i) are the Cartesian coordinates of the p_i. Finally, the optimization is achieved by minimizing the cost of Rectilinear Steiner Minimal Tree Furthermore, this paper also provides a mathematical proof that LL, 22/3, where L, is the length of the Steiner minimal tree and L, is the length of the minimal spanning tree.

- Limitations of work:

  This paper is a fundamental theoretical paper explaining the phenomena and mathematics related with the Steiner Tree Optimization for global routing and does not have any implicit or explicit drawbacks.

On Steiner minimal trees with rectilinear distance

- Author Name: F. K. Hwang
- Year of Publication: 1976
- Work Summary:

We consider Steiner minimal trees in the plane with rectilinear distance. The rectilinear distance d(p,, p2) between two points Pl, P2 iS |Xl X21 + IYI Y21, where the (x,, y,) are the Cartesian coordinates of the p,. For a given finite set P of points, let 15 denote the length of a Steiner minimal tree and 1,. the length of a minimal spanning tree. The main result of the memorandum is that 1i/li> 2. It also provides subsequent mathematical proofs.

- Limitations of work:

  Mathematical proofs of established results have been provided, making it a fundamental conceptual paper. It does not possess any implicit drawbacks.

3.2 Greedy Routing

FastRoute: A Step to Integrate Global Routing into Placement

- Author Name: M. Pan and C. Chu
- Year of Publication: 2006
- Work Summary:

  The placement has become a critical step in the IC design flow. In order to get accurate interconnect information during the placement process, it is desirable to incorporate global routing into it. In traditional global routing approaches, congestion is not considered during Steiner tree construction. So they have to rely on the time-consuming maze routing technique to eliminate routing congestion. Different from traditional approaches, FastRoute proposes a congestion-driven Steiner tree topology generation technique and an edge shifting technique to determine the good Steiner tree topologies and Steiner node positions. Based on the congestion-driven Steiner trees, maze routing can be applied to a small percentage of the two-pin nets once to obtain high-quality global routing solutions. This could dramatically improve the placement solution quality because accurate interconnect information becomes available during the placement process.

  FastRoute can achieve superior quality and speed because of the following techniques.

  1. A congestion-driven Steiner tree topology construction method to distribute routing demand according to the congestion map.
  2. An edge shifting technique to move the horizontal or vertical tree edges in a Steiner tree from congested regions to less congested regions without changing wire length of the tree.
  3. A new cost function based on the logistic function to direct the maze routing to find less congested paths.

- Limitations of work:

  Being a greedy algorithm the only drawback is that if the search space and number of terminal points becomes large then it becomes difficult to find the best solution exhaustively.

FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design

- Author Name: C. Chu and Y. Wong
- Year of Publication: 2008
- Work Summary:

  FLUTE is a very fast and accurate rectilinear Steiner minimal tree (RSMT) algorithm. FLUTE is based on a pre-computed lookup table to make RSMT construction very fast and very accurate for low-degree1 nets. For high-degree nets, a net breaking technique is proposed to reduce the net size until the table can be used. A scheme is also presented to allow users to control the tradeoff between accuracy and runtime. FLUTE is optimal for low-degree nets (up to degree 9) and is still very accurate for nets up to degree 30. So it is particularly suitable for VLSI applications in which most nets have a degree 30 or less. FLUTE with default accuracy is more accurate than the Batched 1-Steiner heuristic and is almost as fast as a very efficient implementation of Prim's rectilinear minimum spanning tree (RMST) algorithm.

- Limitations of work:

  Its performance degrades with increase in the search space dimension and number of terminal points.

3.3 Metaheuristic Optimization Technique

A new swarm intelligence optimizer for robot path planning

- Author Name: H. Duan and P. Qiao
- Year of Publication: 2014
- Work Summary:

  This paper presents a novel swarm intelligence optimizer i.e. Pigeon-Inspired Optimization (PIO) and describes how this algorithm was applied to solve air robot path planning problems. The paper provides a mathematical model and a detailed implementation process of PIO. Pigeon Inspired Optimization works in 2 stages. In the first stage, the pigeon uses the map and compass operator in order to gauge its position with respect to its destination by using magneto reception to shape the map in their brains to fly to its point of disembarkation. In the second part, the pigeon relies on the neighboring landmarks or it will follow the pigeons familiar with the landmarks. Furthermore, in the landmark operator half of the worst performing pigeons are screened out from the swarm as this algorithm tends to focus on the well performing pigeons rather with respect to fitness. The effectiveness and robustness of the proposed PIO algorithm are shown in this paper. The computational results presented here also show that the proposed PIO algorithm can effectively improve the convergence speed, and the superiority of global search is also verified in various cases.

- Limitations of work:

  This paper proposes the algorithm which forms the fundamental inspiration of this work. This paper has no implicit or explicit disadvantages.

Performance Comparison of PSO and Its New Variants in the Context of VLSI Global Routing

- Author Name: S. Nath, J. K. Sing and S. K. Sarkar
- Year of Publication: 2018
- Work Summary:

This paper presents a detailed survey about the Particle Swarm Optimization and its variants. Particle Swarm Optimization (PSO) is a popular meta-heuristic algorithm developed by Dr. Eberhart and Dr. Kennedy being inspired by the bird-flocking and fish-schooling phenomenon. The algorithm keeps track of the best global fitness of all the particles in the swarm as well as the best fitness achieved by the particle in the swarm in a problem hyperspace.

The particle's velocity and position are updated using the following equation:

$$V_i^{k+1} = V_i^k + c_1 rand_1(0,1) \times (pbest_i - s_i^k) + c_2 rand_2(0,1) \times gbest - s_i^k \quad (1)$$

$$S_i^{k+1} = S_i^k + V_i^{k+1} \quad (2)$$

The above equations fail to give a result because of premature convergence of PSO. So to control the velocity of the particle, 2 variants were introduced, namely, the Constriction Factor-PSO (CPSO) and the Inertia Weight- PSO (WPSO). These variants control the velocity of the particle at each iteration so that the position of the particle does not go out of the search space.

The equations for the velocity for CPSO and WPSO are as follows:

$$V_i^{k+1} = k(V_i^k + c_1 rand_1(0,1) \times (pbest_i - s_i^k) + c_2 rand_2(0,1) \times (gbest_i - s_i^k)) \quad (3)$$

$$V_i^{k+1} = w(V_i^k + c_1 rand_1(0,1) \times (pbest_i - s_i^k) + c_2 rand_2(0,1) \times (gbest_i - s_i^k)) \quad (4)$$

Here k = 0.729 and w is calculated using the following equation:

$$w = \left[3 - exp\left(-\frac{s}{200}\right) + \left(\frac{R}{8} * D\right)^2\right]^{-1} \quad (5)$$

where s is the size of the population, R is the relative quality of the corresponding solution normalized between [0, 1] and D is the dimension of the search space.

- Limitations of work:

This paper mainly focuses on the PSO algorithm and its variants which have a disadvantage of premature convergence.

CHAPTER 4. PROBLEM FORMULATION

4.1 Problem Statement

Given a set T of n points, called terminals, in the plane, find a set S of additional points, called Steiner points, such that the length of a rectilinear minimum spanning tree of T is minimized. Furthermore, the main objective is to reduce the error ratio i.e. the ratio of the weight of Rectilinear Steiner Minimal Tree to the weight of the Rectilinear Minimal Spanning Tree.

CHAPTER 5. PROPOSED WORK

5.1 Proposed Methodology

Algorithm to solve Rectilinear Steiner Minimal Tree Problem

In this work, we have endeavored to solve the Rectilinear Steiner Minimal Tree problem using a dynamic algorithm, namely Pigeon Inspired Optimization Algorithm.

Pigeon Inspired Optimization:

Pigeon Inspired Optimization is developed from the homing skills of pigeons where each pigeon uses the magnetic field of the earth to find their position and ultimately converge to the best position. The PIO works using two operators

1. The Map and Compass Operator:

In this case, the rules are defined with position Xi and the velocity Vi of each Pigeon i , the positions and velocities are updated in each iteration t as follows:

$$Vi\,(t) = Vi\,(t-1)\,.e^{-Rt} + rand.(Xg - Xi\,(t-1)) \quad (1)$$

$$Xi\,(t) = Xi\,(t-1) + Vi(t) \quad (2)$$

Here, R is the map and compass operator, Xg is the current global best position acquired by the swarm of pigeons and rand is a random number.

2. The Landmark Operator:

In this operator, half the number of pigeons is decreased at each step by Np . But the pigeons are unfamiliar with the surroundings apart from being far away from the destination. Let Xc (t) be the centre of some pigeon's position at time t and assuming that every pigeon can fly straight to the destination the position update rule for pigeon i at tth iteration is given by the following set of equations:

$$Np\,(t) = Np(t-1)/2 \quad (3)$$

$$Xc\,(t) = \Sigma\, Xi(t).fitness(Xi(t)) \,/\, Np\, \Sigma\, fitness(Xi(t)) \quad (4)$$

$$Xi(t) = Xi\,(t-1) + rand \times (\,Xc\,(t) - Xi(t-1)) \quad (5)$$

5.2 Proposed Algorithm

Initially, the Steiner sets are randomly initialized within the search space. Then the coordinates of the Steiner points are fed to the PIO algorithm along with the coordinates of the Terminal points. The PIO algorithm checks the fitness of the

arrangement of each of the Steiner sets along with the terminal points. The algorithm then uses the map and compass operator along with the landmark operator to optimise the arrangements of the Steiner sets which tends to maximize the fitness while minimizing the objective function cost. The objective function used in this work is the cost of the Minimal Spanning Tree of the complete arrangement. On the other hand, fitness is the reciprocal of the objective function. Furthermore, due to the usage of the map and compass and the landmark operator, the PIO algorithm behaves consistently with higher accuracy. Additionally, it also makes the algorithm robust and reduces the overall execution time.

Pigeon Inspired Optimization Algorithm:

I. Map and Compass

Map_and_Compass(G(V, E), $iter_{max}$, pigeons):

1.while iter < $iter_{max}$ do
2. for each pigeon p in pigeons do
3. fitness(p) = 1/ MST (p)
4. g_{best} = max(fitness)
5. for each pigeon p in pigeons do
6. $term_1 = c_1 \times rand_1(0, 1) \times (p_{best} - p.S_i^k)$
7. $term_2 = c_2 \times rand_2(0, 1) \times (g_{best} - p.S_i^k)$
8. $p.V_i^{k+1} = wV_i^k + term_1 + term_2$
9. $p.S_i^{k+1} = p.S_i^k + p.V_i^{k+1}$
10. iter = iter + 1

II. Landmark Operator

Landmark_Operator(G(V, E), pigeons, N_{pmax1} , N_{pmax2}):

1. $N_p = N_{pmax1}$
2. while $N_p > N_{pmax2}$ do
3. pigeons = pigeons[0 : N_{pmax1}]
4. sort pigeons based on fitness
5. $X_c = \Sigma\ p.X_i(t) \times$ fitness$(p.X_i(t))$ / $N_p \Sigma$ fitness$(p.X(t))$
6. $X_i(t) = X_i(t-1) + rand \times (X_c(t) - X_i(t-1))$
7. $N_p = N_p / 2$
8. End while

CHAPTER 6. EXPERIMENTS AND ANALYSIS

6.1 Experimental Setup and Attributes

All the algorithms were executed on a system running on a 64-bit Ubuntu 18.04.3 LTS as the OS. The device consisted of DDR3 RAM of 8GB. The processor used by the system was a quad-core AMD A6-7310 APU with a processing speed of 2 - 2.4 GHz. All the algorithms were coded in Python 3.6 and its scientific library, namely, Numpy. Visualization was done using Python plotting library, namely, Matplotlib. The simulation setup uses predefined datasets generated within the search space for 10, 20, 30, 40 and 50 terminal points respectively. The individual dataset includes the X and Y coordinates of the respective terminal points represented as a coordinate list within a Numpy file (.npy). All the algorithms were executed on the predefined dataset. It needs to be noted that no simulation softwares was used in this work. The algorithm was executed in python terminal and the results were collected and visualized in the standard runtime environment.

Table I provides the details of the fundamental parameters that were initialized for running the experiment. These parameters include the total number of iterations for which each algorithm was executed, the total number of particles instantiated and the total number of Steiner populations per particle. These parameters are kept constant for all the algorithms throughout the experiment.

TABLE I. GENERAL INITIALIZATION PARAMETERS

Algorithm	Number of Iterations	Number of Particles	Population per Particle
PIO	15	150	(Total number of terminal points) - 2
PSO			
CPSO			
WPSO			
Dimension of Search Space			500 X 500

6.2 Experimental Results.

The results obtained through the experimental analysis are provided in the following tables. Table II, Table III, Table IV, Table V and Table VI contains the results of Pigeon Inspired Optimisation, Particle Swarm Optimisation, Constricted Particle Swarm Optimisation and Weighted Particle Swarm Optimisation for 10, 20, 30, 40 and 50 terminal points respectively.

TABLE II. DATA CHART FOR NO. OF TERMINAL POINTS = 10, SEARCH SPACE DIMENSION = 500 X 500, NUMBER OF ITERATIONS = 15

Algorithm	Number of Steiner Points	Numbe r of Trials	Total Weight of RMST	Total Weight of RSMT	Error Ratio	Execution Time (s)
PIO	2	100	1260	1167	0.9261	4.086
PSO	3	100	1260	1179	0.9357	5.709
CPSO	2	100	1260	1251	0.9929	4.560
WPSO	3	100	1260	1169	0.9278	6.536

TABLE III. DATA CHART FOR NO. OF TERMINAL POINTS = 20, SEARCH SPACE DIMENSION = 500 X 500, NUMBER OF ITERATIONS = 15

Algorithm	Number of Steiner Points	Number of Trials	Total Weight of RMST	Total Weight of RSMT	Error Ratio	Execution Time (s)
PIO	3	100	2105	1971	0.9363	37.046
PSO	2	100	2105	2002	0.9511	47.846
CPSO	3	100	2105	2058	0.9777	48.487
WPSO	3	100	2105	2021	0.9601	46.354

TABLE IV. DATA CHART FOR NO. OF TERMINAL POINTS = 30, SEARCH SPACE DIMENSION = 500 X 500, NUMBER OF ITERATIONS = 15

Algorithm	Number of Steiner Points	Number of Trials	Total Weight of RMST	Total Weight of RSMT	Error Ratio	Execution Time (s)
PIO	2	100	2071	2038	0.9841	59.475
PSO	1	100	2071	2048	0.9889	103.095
CPSO	0	100	2071	2071	1.0	77.578
WPSO	1	100	2071	2035	0.9826	97.696

TABLE V. DATA CHART FOR NO. OF TERMINAL POINTS = 40, SEARCH SPACE DIMENSION = 500 X 500, NUMBER OF ITERATIONS = 15

Algorithm	Number of Steiner Points	Number of Trials	Total Weight of RMST	Total Weight	Error Ratio	Execution Time (s)

				of RSMT		
PIO	3	50	2751	2715	0.9869	179.136
PSO	3	50	2751	2705	0.9833	470.857
CPSO	0	50	2751	2751	1.0	351.250
WPSO	3	50	2751	2749	0.9992	378.257

TABLE VI. DATA CHART FOR NO. OF TERMINAL POINTS = 50, SEARCH SPACE DIMENSION = 500 X 500, NUMBER OF ITERATIONS = 15

Algorithm	Number of Steiner Points	Number of Trials	Total Weight of RMST	Total Weight of RSMT	Error Ratio	Execution Time (s)
PIO	1	30	2831	2811	0.9929	469.832
PSO	0	30	2831	2831	1.0	1128.945
CPSO	0	30	2831	2831	1.0	816.897
WPSO	0	30	2831	2831	1.0	1194.963

The RSMT generated by the PIO algorithm on a set of 50 terminal points is provided in Fig. 3. Fig. 4 represents the RSMT generated by the PSO, CPSO and WPSO algorithm on the same set of 50 terminal points for 15 iterations with the total number of trials being 30. These figures provide the graphical results which are based on the experimental results presented in Table VI.

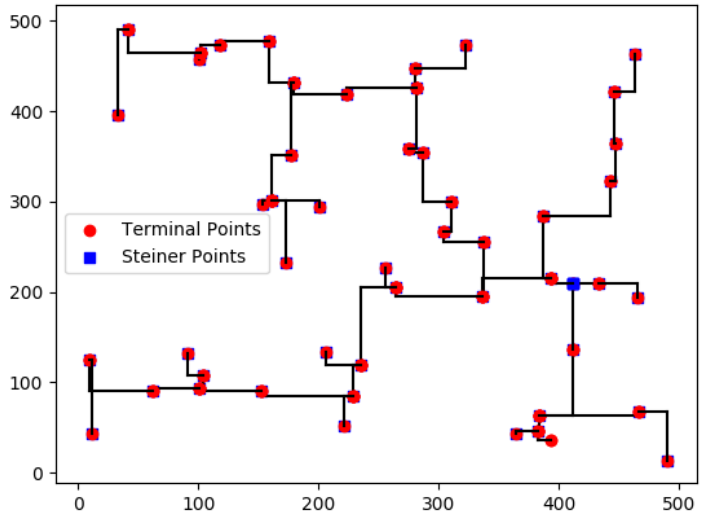

Fig 9 : The solution of PIO Algorithm for 50 Terminal Points

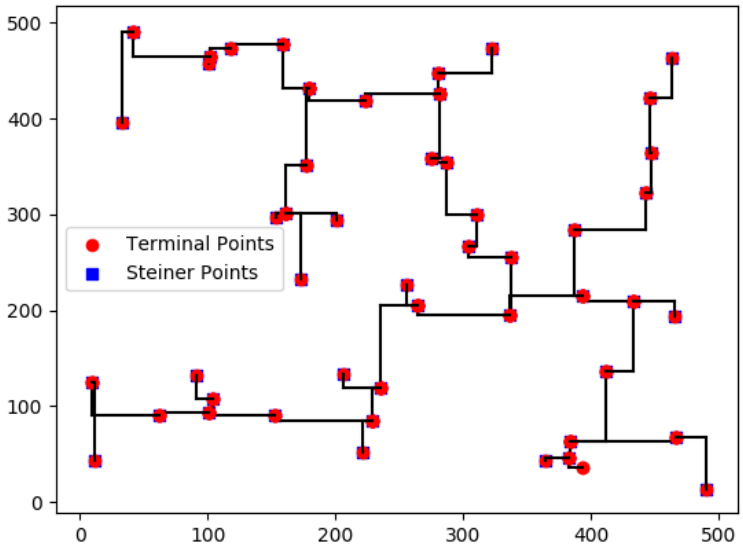

Fig 10 :The solution of PSO, CPSO and WPSO

Algorithm for 50 Terminal Points

6.3 Result Analysis and Discussion

The comparative analysis presented in the above tables is based on the search space dimension, the number of terminal points and Steiner points that could be placed by the different metaheuristic algorithms. The comparison also takes into consideration the total weight of the Rectilinear Minimal Spanning Tree (RMST) without the Steiner points along with the total weight of the Rectilinear Steiner Minimal Tree (RSMT) by considering the Steiner points along with the terminal points.

The number of trials is kept constant which represents the total number of times each optimisation algorithm was triggered. The result reported is the best result found among all the trials. The reported result also contrasts between the execution time that the individual algorithm requires to descend to the optimal solution. Finally, the error ratio represents the relation between the total weight of RMST and the total weight of the RSMT and can be defined by the following equation:

$$\text{Error Ratio}=(\text{Total Weight of RSMT})/(\text{Total Weight of RMST}) \quad (11)$$

It can be easily observed from the reported results that the PIO algorithm has the least execution time amongst all other algorithms demonstrated in the experiment. Furthermore, it can be also perceived that the PIO algorithm maintains a high and consistent accuracy in comparison to the PSO and its variants. It is evident from the results reported in Table VI that when the other metaheuristic algorithms failed to place a Steiner point within the search space, PIO proved to be successful in placing at least one Steiner

point within the search space in order to reduce the total weight of the minimal spanning tree.

CHAPTER 7. CONCLUSION

This work focuses on optimizing the global routing problem with respect to the cost of routing in terms of energy loss which increases proportionately with the length of the routing path. The proposed work tends to optimize the global routing problem through the Steiner Tree Optimization problem using the Pigeon Inspired Optimization algorithm. The experimental results show that the performance of PIO is better than PSO. Moreover, PIO is also competent in comparison with the PSO with respect to execution time while maintaining its high efficiency and accuracy in placing the Steiner points within the search space. The PIO algorithm also has a faster rate of convergence in contrast to the PSO.

REFERENCES

[1] M. Hanan, "On Steiner's problem with rectilinear distance", SIAM Journal on Applied Mathematics, vol. 14, no. 2, pp. 255-265, 1966.

[2] F. K. Hwang, "On Steiner minimal trees with rectilinear distance", SIAM Journal on Applied Mathematics, vol. 30, no. 1, pp. 104-114, 1976.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization", In Proc. of IEEE International Conference on Neural Networks, pp. 1942-1948, 1995.

[4] S. Rajagopalan and V. V. Vazirani, "On bidirected cut relaxation for the metric Steiner tree problem", In Proc. the tenth annual ACM-SIAM symposium on Discrete algorithms, pp. 742-571, 1999.

[5] Y. Shi and R. C. Eberhart, "Empirical Study of Particle Swarm Optimization", IEEE Congress on Evolutionary Computation, pp. 1945-1950, 1999.

[6] G. Robins and A. Zelikovsky, "Improved Steiner Tree Approximation in Graphs", SODA, Pp. 770-779, 2000.

[7] M. Clerc and J. Kennedy, “The particle swarm - explosion, stability and convergence in a multidimensional complex space,” IEEE Transactions on Evolutionary Computation, vol. 6, no. 1, pp. 58-73, 2002.

[8] M. Pan and C. Chu, "FastRoute: A Step to Integrate Global Routing into Placement," 2006 IEEE/ACM International Conference on Computer Aided Design, San Jose, CA, 2006, pp. 464-471, doi: 10.1109/ICCAD.2006.320159.

[9] S. Tsuda, K. P. Zauner, and Y.-P. Gunji, "Robot control with biological cells," Biosystems, vol. 87, no. 2-3, pp. 215-223, 2007.

[10] C. Chu and Y. Wong, "FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 1, pp. 70-83, Jan. 2008, doi: 10.1109/TCAD.2007.907068.

[11] W.-L. Zhong, J. Huang and J. Zhang, "A Novel Particle Swarm Optimization for the Steiner Tree Problem in Graphs", IEEE Congress on Evolutionary Computation, pp. 2460-2467, 2008.

[12] T. Arora and M. E. Moses, "Ant colony optimization for power efficient routing in Manhattan and non-Manhattan VLSI architectures", Swarm Intelligence Symposium, IEEE, Pp. 137-144, 2009.

[13] A. Tero, T. Nakagaki, K. Toyabe, K. Yumiki, and R. Kobayashi, "Amethod inspired by Physarum for solving the Steiner problem," Int. J. Unconvent. Comput., vol. 6, no. 1, pp. 109 123, 2010.

[14] H. Duan and P. Qiao, "Pigeon-inspired Optimization: A new swarm intelligence optimizer for robot path planning", International Journal of Intelligent Computing and Cybernetics, pp.24-37, 2014.

[15] M. Caleffi, I. F. Akyildiz and 1. Paura, "On the Solution of the Steiner Tree NP-Hard problem via Physarum BioNetwork," IEEE/ACM Transactions on Networking, vol. 23, no. 4, pp. 1092-1096, 2015.

[16] S.Nath, S.Ghosh and S.K.Sarkar "A Novel Approach to Discrete Particle Swarm Optimization for Efficient Routing in VLSI Design" IEEE Proceedings of 4th International Conference on

Reliability, Infocom Technologies and Optimization, 2015, ISBN978-1-4673 7230-5.
[17] S. Ghosh, S. Nath, R. Biswas, P. Venkateswaran, J. K. Sing, S. K. Sarkar, "PSO Variants and its Comparison with Firefly Algorithm in Solving VLSI Global Routing Problem", 2018 IEEE Electron Device Kolkata Conference, EDKCON, 2018.
[18] S. Nath, J. K. Sing and S. K. Sarkar, "Performance Comparison of PSO and Its New Variants in the Context of VLSI Global Routing", IntechOpen, pp. 1-21 of 21, 2018.
[19] S. Nath, P. P. G. Neogi, J. K. Sing, S. K. Sarkar, “VLSI Routing Optimization based on modified Constricted PSO with Iterative RLC Delay model", 2018 International Conference on Current Trends towards Converging Technologies, ICCTCT, 2018.