A NOVEL SEARCH ALGORITHM FOR COLUMN BASED SPATIAL DATABASE

A thesis submitted to the University of
Petroleum and Energy Studies

For the Award of

DOCTOR OF PHILOSOPHY

In

Computer Science Engineering

By BHAGWANT SINGH

July 2020

SUPERVISOR

Dr. Kingshuk Srivastava

CO-SUPERVISOR

Dr. Dharmendra.K Gupta

UPES
UNIVERSITY WITH A PURPOSE

Department of Informatics

School Of Computer Science

UNIVERSITY OF PETROLEUM & ENERGY STUDIES,

A NOVEL SEARCH ALGORITHM FOR COLUMN BASED SPATIAL DATABASE

A thesis submitted to the University of
Petroleum and Energy Studies

For the Award of

DOCTOR OF PHILOSOPHY

In

Computer Science Engineering

By

BHAGWANT SINGH

(SAP ID: 500031709) July 2020

SUPERVISOR
Dr. Kingshuk Srivastava

Assistant Professor-SG SoCS,
UPES, Dehradun, Uttarakhand

CO-SUPERVISOR
Dr. Dharmendra.K Gupta
Professor SOE,

UPES, Dehradun, Uttarakhand

Department of Informatics

School Of Computer Science

UNIVERSITY OF PETROLEUM & ENERGY STUDIES,

DECLARATION BY SCHOLAR

I hereby declare that this submission is my own work and that to the best of my knowledge and belief. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or another institute of higher learning, except where due acknowledgment has been made in the text.

[illegible]

BHAGWANT SINGH

Date:

Place: Dehradun

UPES
UNIVERSITY WITH A PURPOSE

Great Place To Work® Certified MAR 2020-FEB 2021 INDIA

THESIS COMPLETION CERTIFICATE

This is to certify that the thesis on **"A Novel Search algorithm in Column based Spatial Database"** submitted by **Bhagwant Singh** to School of Computer Science and Engineering, University of Petroleum and Energy Studies (U.P.E.S), Dehradun in Partial completion of the requirements for the award of the Degree of Doctor of Philosophy (Engineering- Computer Science) is an original work carried out by him under our joint supervision and guidance.

It is further certified that the work has not been submitted anywhere else for the award of any other diploma or degree of this or any other University.

[illegible]

Dr. Kingshuk Srivastava
(Supervisor)

[illegible] 12/07/2021

Dr. Dharmendra.K Gupta
(Co-Supervisor)

ABSTRACT

The Conventional technologies designed in the beginning of the 21st century thrived mostly on the concept of Rich-Data/Multimedia Data. These data in consideration is acquired from various sources most of which are mobile devices as well as sensors and other TPS (transaction processing system) systems. These type of data is handled mostly in few types of storage systems namely Date Warehouses, Clouds and a newer addition called Data Lake. One of the major requirement for every dataset is location tagging which was non-existent a few years back, but has become imperative for any type of contemporary analytical process. So time stamping and location tagging of every dataset has become paramount in its importance today.

The degree and depth of the metadata has heightened the cognizance among the researcher for advancement in infrastructure designed for handling spatial data. “Spatial Big Data (SBD)” has evolved with evolving V’s of geographical data. Spatial big data has bolstered the significance of Geographic Information Systems (GIS) in multiple mainstream community applications for i.e. intelligent route planning for aircraft, Social Media analysis, Transportation and Navigation system, Disaster management, Urban planning & monitoring, and Health hazards etc. Increasing influx of user-generated data over open community has amplified the challenges faced by the ACID based management system in handling SBD. This transition from ACID to BASE theorem has encouraged the analyst to work with NoSQL based systems for the processing of SBD.

Storage, search and retrieval are the three main functionality of any database management system. Column based data fetch and index model (CBDFI) present a novel algorithm to search spatial big data stored in column database. Proposed algorithm incorporates innovative location code and inventive hybrid indexer that can change the outlook of the researcher working with location aware dataset. The current research compares IL-Quadtree TopK-SK, S2I-Quadtree TopK-Sk, and I3-Quadtree TopK-SK search algorithm and presents a novel CBDFI search algorithm in comparison to the after mentioned algorithms.

CBDFI model present a unique location code “hs code” which converts and presents any sexagesimal location into 7 bit code. The model achieves to formulate “hs code” by calculating the weighted ASCII of each location number and dividing them with prime factor belonging to p alpha {} superset. This proposed method can generate and retrace location without any backtracking error. The “hs code” is more compressed then its contemporary Geohash as it is string of 7 bit whereas latter is string of 9 bit. Hidden markov model present a probability based mathematical model to traverse from one hs code to another. This probability model formulate the base line for searching any spatial entity in CBDFI model.

CBDFI model present singleton indexing data structure that can be used for both textual and spatial indexing hence ensuring the novelty and uniqueness of the purposed research. CBDFI model generate Hash inverted file to index the keyword in the any SBD. The indexer presents approximately one-sixth percentage improvement in the execution time with increasing keywords (above 20000) therefore ascertain its utility in the field of SBD. CBDFI model utilize Hs-I Quadtree, where each child node is a cluster of 24 different location hence improving the performance by 24 folds.

CBDFI model incorporate the concept of Pre-fetch table to further improve the performance of the search algorithms. The keyword in any mentioned top k spatial keyword search query is compared for most frequent keyword list, global probable keyword list and local probable keyword list for positive match. Finally, CBDFI model is designed for Industry 4.0 where the improved fetch time of the presented search mechanism will present advance solution for location aware services.

Keywords: Spatial Big data, hs code, spatial indexer and Top-K search.

ACKNOWLEDGMENT

First and foremost, my all devotions are to my almighty for his countless blessings. My unrestrained appreciation goes to my advisors; Dr. Kingshuk Srivastava and Dr. D.K gupta for all the help and support they have extended to me throughout the course of this work. I simply cannot imagine how things would have proceeded without their help, support, patience, and contributions. I also extend my special acknowledgment to Dr. Manish Prateek, Director, SoCSE, UPES, India for his valuable time in consultations and discussions. I also like to thank Dr. Manas Ranjan Pradhan and Dr. Venkatardi Meriboina who provided insight and expertise on search mechanism and their performance parameters which greatly assisted the research.

I also acknowledge my many colleagues and friends as I had a pleasant, enjoyable and fruitful company with them. I am also extremely grateful to the celebrated authors whose noteworthy works have been consulted and referred in my research work. Finally, I wish to express my gratitude to my family member’s especially my wife for being patient with me and offering words of encouragement to spur my spirit at moments of depression.

Bhagwant Singh

TABLE OF CONTENT

LIST OF ABBREVIATIONS

ASCII	American Standard Code of Information Exchange
ACID	Atomicity, Consistency, Isolation, Durability
BASE	Basically Available, Soft state, Eventual Consistent.
CAP	Consistency, Availability, and Partition tolerance
CEP	Complex event Processing
CBDFI	Column Based Data Fetch and Index
GIS	Geographic Information System
GPS	Global Positioning system
GDB	Geo-Database
HMM	Hidden Markov Model
Hs	Hexa Spatial
Hy-S	Hybrid Spatial
Hs-I	Hash Spatial Inverted
k-NN	k- Nearest Neighborhood
MBR	Minimum bounded Rectangle
NoSQL	Not SQL
OGC	Open Geospatial Consortium
PD	Projected Distance
R-DBMS	Relational Database Management System
SBD	Spatial Big Data
UAV	Unmounted Aerial Vehicle
UTM	Universal Transversal Mercator
VGI	Volunteered Geographic Information

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CHAPTER 1

PARADIGM SHIFT OF SPATIAL DATA

1.1 INTRODUCTION

The 21st century is computing data in petabytes and terabytes. Managing spatial information with accumulative data, results in complex processing and analysis. To provide tool with the capabilities of handling hetro-type spatial data is a necessity and such data pose the problems of 1) Data Quality and 2) Ownership of Data [Johnson (2016)]. For optimal storing of spatial data, data warehouse is a concept but each and every version of the information extracted from it doesn't carry the key points about the pervious extraction and the advancement to be conducted in the future. This results in solution for a particular time window but correlating with other solution at different time and place turns into a nightmare.

Now a day, advance technologies are rapidly and dynamically generating data which is readily available for analysis. Engendered data from these devices can capture an instance with similar intrinsic behavior but registered at diverse location. These non-traditional dataset are majorly unstructured & semi-structured in nature , hence in the contemporary sprouting data analyzing scenario these characteristics turn out to be the biggest concern of analysisists. Industry 4.0 is registering progressive technologies, which are engendering humongous amount of data at exceptional rate (figure1-1) for further analysis. Various organizations identify information with intrinsic location element, relevant for decision making [Giuliani et.al. (2011)]. Place complexity of such data can be abridged by introducing location as metadata, hence exponential increase in data size can be encountered. Eighty percentent of the data is geotag therefore registering spatial data a necessity [Schade et.al. (2015)]. Studies have shown that if text + location can recuperate search query. Latest research shows that 12% of the quires searched over web have place name [Barewar et.al. (2014)].

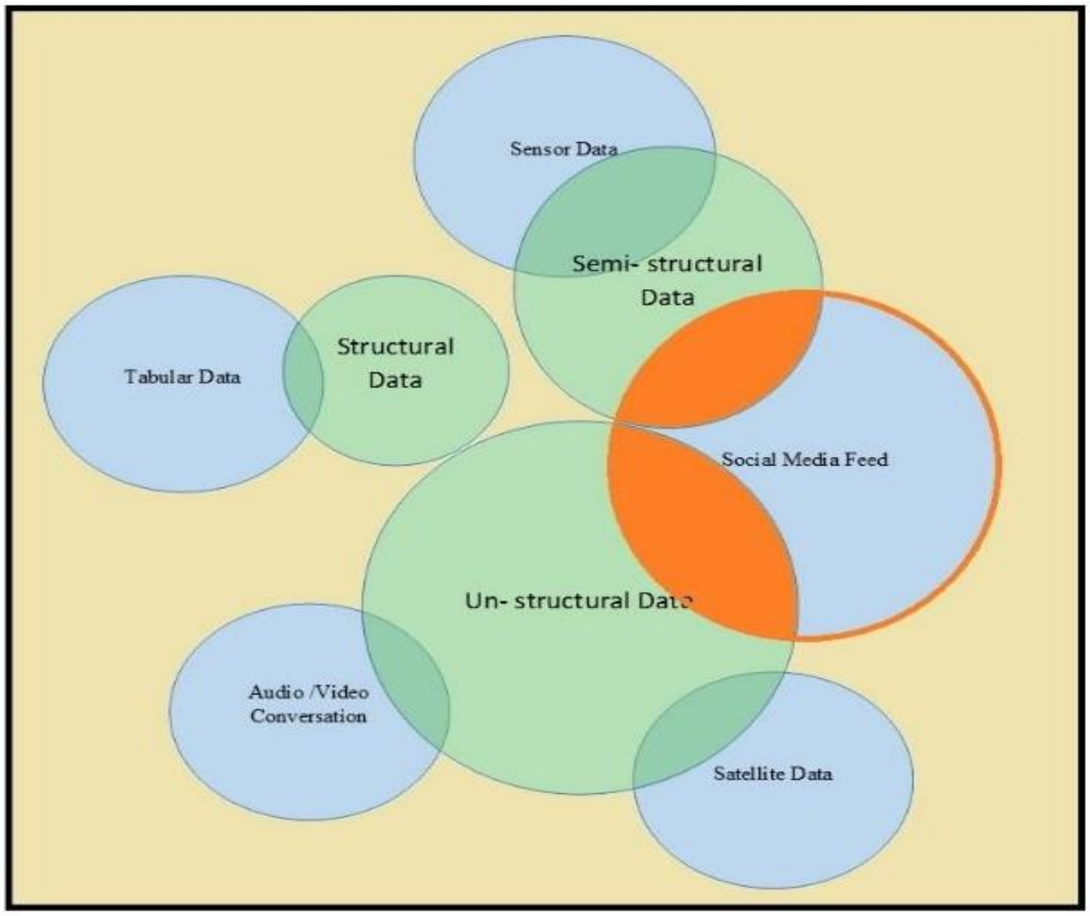


Fig 1-1: Different form of Data

Geographic Information System (GIS); serve as an infrastructural backbone for capturing storing and analyzing location intrinsic data. The evolved field of cartography, bought GIS into the limelight. The government of Canada played reformed government initiatives but utilizing the visualization techniques of GIS. One such initiative was taken by Roger Tomlinson, an English geographer in mid-1960. Jack Dangermond founder E.S.R.I (1969), is a leading commercial firm which mostly focused on developing GIS application for government and commercial sectors. Concomitant technologies of GIS viz. global positioning systems (GPS) and Radio Frequency Identification Tags (RFID) was expended during Cold War U.S.A.

Towards the end of 19th century manifold advancements to enhance technical support in GIS computational systems has enabled multiple GIS related software, based on the principle of enabling spatial analysis to be financially feasible commodity in competitive market.

Any relational database can be used to store data of such characteristics but storing spatial information result in explicitly defining relationships which result in slower processing. Geodatabase provide solution to this problem. Geodatabase is built on the concepts of off-the-shelf database with advance application logic layer which help in storing spatial information and implicitly generating behavioral relationship. Geodatabase data model have rich collection of objects and features and these features can be store in four dimensions (x, y. z and m). M dimensions help in providing metadata for the data and help in bridging the gap between past and future. The feature of versioning and long transaction help to fetch the data from different location and facilitate multiple user access.

About 90% of data which are being generated today is unstructured in nature and this has become a challenge in the sector of analytics. To handle unstructured data the most prominent and pioneering tool is Hadoop. Many proprietary tools launched in market such as "IBM Info-Sphere Big Insight" and "MS SQL Server Big Insight" is also incorporating Hadoop as base architecture. Hadoop has irreplaceable advantage in scalability, robustness, calculated performance and cost effectiveness with low end hardware support and has become the main big data analysis platform in the current market. Different kriging algorithm can be applied over Hadoop to provide solution for spatial estimation or interpolation.

Database management systems have been leading storage technology in market for decades. One has seen the evolution of database from simple files system to object oriented relation database management systems. Data vendors are constantly improving the database management systems but R-DBMS still struggling with large volume of data [Mohan (2013)]. Over the decade every development in database results in more and more level of abstraction of data. ACID theorem is used to maintain data consistency, however when volume of data is in petabytes the speed of retrieval of information decelerate. Hence the concept of de-normalization is luxuriated; it is perceived that for better analysis data should be in 1st normal form. For this a contemporary technology, NoSQL (not only SQL) databases is introduced. NoSQL used non-transactional approach called BASE and is able to support larger volumes of data by providing faster data access and cost savings.

Location pairing is an important and inherent obligation of every service provider worldwide. Taking in incalculability of the engendered spatial data at an inimitable rate, probing information turn out to be a decisive job. Diverse web/mobile podiums such as online directories (yellow pages), social media sites (twitter, Facebook, flicker), web crawlers

(Google, Yahoo, Bing), and GPS enables devices (airline, taxi, rail) provide semantic location aware data of the user. Perception of analogous data with intrinsic location beheld under different context deliver unlikely fallouts. Hence encouraging the use of advanced and improved spatial temporal data processing tools are highly suggested. Location aware services tends to record trajectory of the object to understand cohesive nature of the object in consideration. For that reason, location get associated with the objects, its different interaction with environment and review about the activities in the neighborhood. Location and characteristic keywords can be paired together and research ensuring effective search of information has been undertaken during the last decade [Hong et al. (2017)] [Zhange et.al. (2016)][Griffith et.al. (2016)]. Combining spatial search and keyword search has presented hybrid search mechanism to work with current spatial data known as "Spatial keyword Search". Top- K spatial keyword query is one of the most remarkable algorithm used to manipulate existing spatial data. Over a time, advancements in the Top-K search algorithm to cater different problems, but no research optimize the latitude and longitude with associated keyword. Multiple commercial and scientific users are stipulating spatial data to provide innovative solutions. These geospatial data sets range in terabytes and even in petabytes. Storing such volume of data is a costlier affair hence new and improved storage technologies are required to accumulate expanding spatial data. The spatial data is scattered geographically and is of heterogeneous-typed. Therefor search and analysis of spatial data is foremost concern of solution providers.

1.1.1 SPATIAL DATA IN 4 V'S ENVIRONMENT

Spatial data are data that inhabits space. Spatial data conveys topological and/or distance information and spatial indexing and spatial access methods are used for shaping and editing them. "Everything is related to everything else but near things are more related than distant things", served as the essential characteristic of spatial data [Tobler et.al. (1970)]. Based on the first law of Geology, in order to assist spatial dependency, spatial autocorrelation was pioneered [Goodchild (1986)]. The second essential characteristic of spatial data is heterogeneity. Each object in real world plane exhibit distinctive characteristics, hence revealing that not all spatial data exhibit stationary characteristics [Barewar et.al. (2014)]. Incomparable fuzzy nature serve as the third and the last characteristic of spatial. These characteristic of spatial data impersonates various challenges for when the storage and manipulation of such data is taken in consideration.

Big data is term used to define voluminous datasets measured in petabytes or terabytes. Big data can be of any type's structured, semi-structured and unstructured type. Any data that has the potential to be mined for information can be categorized as Big Data. It is so huge and complex that analysis, capturing, curation, retrieving, storing, transferring, and visualizing of such a data turn into nightmare.

Hadoop is a java based framework designed for disseminated processing of large data set athwart cluster of computers. It was designed to provide fast processing by detecting and handling failures of computers at the application layer itself. During the last few years Hadoop has become the mainstream software framework for Big Data processing. Major operations on spatial data require global indexing but many of existing big data system for graph data such as MapReduce is designed for relational database [Doulkeridis et.al. (2014)] and lack systematic framework to use and create indexes. To overcome this problem of global indexing, HadoopGIS and SpatialHadoop were introduced.

Geolocation enabled data can be classified as spatial data. Conventional any spatial data or geographical data, can be divided into two parts the attribute data i.e. event's information capturing the details of that event and the spatial data i.e. the location where the event has occurred (figure 1-2).

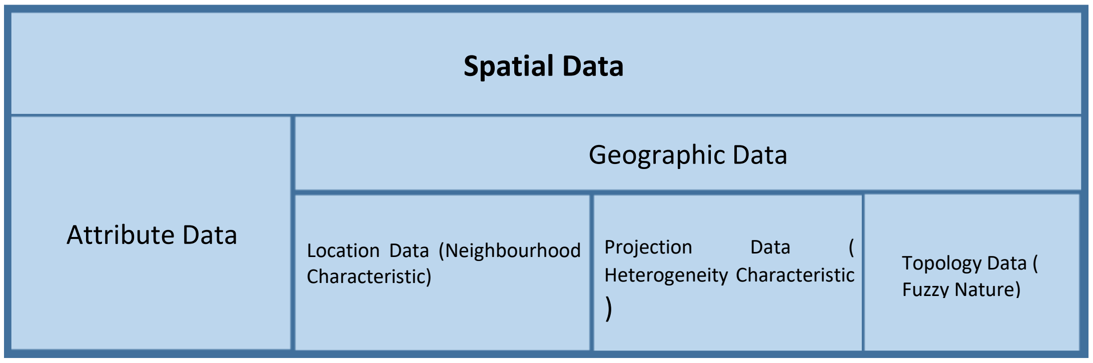

Spatial Data
Geographic Data
Attribute Data
Location Data (Neighbourhood Characteristic)
Projection Data (Heterogeneity Characteristic)
Topology Data (Fuzzy Nature)

Fig. 1-2. Characteristic of Spatial Data

Meer latitude and longitude doesn't complete "location" data. Various other topological information such as projection system used to capture the instance, local and global datum and topology also comprehend to location information, hence handling of spatial data is a complex phenomenon.

The evolution of smart-era has constituent geotagging as a primitive feature of multiple devices. During the first decade of 20th century an alarming influx of data has been generated by different social media platforms [Song et.al. (2014)]. Immense pool of expressed feeling through blogs and tweets, vital information like friend and important engagement and real experiences shared through social media, results in enormous data for processing. Global platform such as social media generate the large real time data over time at fast rate and registering all the different formats of inputs, result in what has created the concept of Spatial Big data. Various example of spatial big data include check-ins [Medina et.al. (2012)] by various user globally at the same time and understanding the epicenter of each check-ins, GPS-tracking of user using smart devices, Unmanned aerial vehicle (UAV)/Wide area motion imagery (WAMI) [Medina et.al. (2012)] video and generating roadmaps from various user generated content, Waze, Open Street Map etc.

Big Data with location as its core property formulate Spatial Big-Data (SBD). Extend of information produced by spatial big data, can be processed and analysis in far-fetched manner hence presenting researcher with new filed of spatial analysis. Justifying captured data as spatial big data depends on the context of data. Micro and macro analysis of the situation can be comprehend by Spatial Big-Data and deliver analyst.

Volume: Different digital device present over the World Wide Web or then globe can accumulate immense for processing and increase the apprehension analysist. Hence the off the rack data management system need to amend their architecture to process such voluminous data.

Velocity: Magnitude of sources worldwide are generating endless data at a very fast rate which support the volume of SBD. This unremitting data generate by various advance technology worldwide at rapid rate is providing velocity to Big-Data.

Variety: Nowadays, anyone can generate data about any entity in consideration. To investigate any event an army of modern deceives are supported by the current technology. Therefore most of the registered data are of the same entity but in different format and are schema less majorly that is unstructured or semi-structured. Large data traveling from different paths for analysis result in the immense variety of Big- Data.

Veracity: One of the most proficient reason for let-down of the conventional system is the messiness of the data available for analysis. Pre-processing capability of the system get highly hampered by the heterogeneous typed nature of data generated.

1.1.2 SPATIAL DATABASE MANAGEMENT SYSTEM

Spatial data management system is the basic functionality of Geographical Information System (GIS. To store such data K-tree data structured is used [Barewar et.al. (2014)], this categorized data object comprise of object class, element class and element dataset. Object class can be considered as non-spatial data set which includes the different characteristic of the geographical data. Whereas element class and element dataset is set which represent geometrical attribute and their respective reference system. To manage spatial data various features such as spatial position, relationship of geographic entity, nomenclature, type and quantity of geographic entity and time features are stored concretely. In spatial database attribute data and graphic data of an object are integrated together. There are two ways to integrate data either to combine spatial and attribute data before integrating or integrate attribute and spatial data separately and then combine them. Various spatial solutions can be provided by both conventional and unconventional databases. MySQL and Postgre-SQL has incorporated OGC's SFS and SFSQL to provide the functionality of spatial analysis, but face the problem with scalability of the current data. To deal with the large scale of unstructured data MongoDB, SpatialHadoop, BigTable and CouchDB etc document based NoSQL based database can be used showcase the concept of geohasing for spatial indexing [Zhang et.al. (2014)].

"The Geodatabase is unique data format that is similar in structure to coverage data model and also include the functionality of multiuser editing". Similar to the relation Dbms, a geodatabase is the collection of geographic dataset. Geographical dataset may include features class, raster data, attribute tables relationship between attribute and relationship between features.

A geodatabase is a pool of geographic datasets of several types arranged under corporate file system folder such as Microsoft Access database, Oracle, Microsoft SQL Server, PostgreSQL, Informix, or IBM DB2. Principle of Object relational model is used to design a geodatabase. All the real time entities are stored in rows in form of objects. Hence it may be said that Geodatabase is off the shell relational Dbms with advance application logic layer on it.

While storing data in geodatabase one use well defined column type structure. Along with using standard data type GDB also used extended spatial datatype. This extended spatial data type is used to store the datum and projection. Geodatabase store dataset rather than storing data. These dataset can be following different type feature class, raster dataset, and attribute table. Schema of GIS data is stored using following main table GDB_Item, GDB_ItemType, GDB_ItemRelationship and GDB_ItemRelationshipType. GIS deals with data which is extremely large in size and will be assessed by large number of users. Key concept of geodatabase is to leverage concepts of R-Dbms and scale it up to meet the requirement of GIS.

1.1.3 SPATIAL TERMS AND DEFINATION

The proposed research highlight various different concepts of spatial data. The definition of some of the term are as follows:

Column Database: A type of non-conventional database that store data in column rather than rows

Geo-Database: A relational database used to store, query and manipulate spatial data. It is also called as Spatial Database

Geo-hashing: It is an encoding technique used to capture the location on the earth surface in form of a varchar string. More the length of the string precise will be the location

Global Probable keywords: The most prominent keyword identified for a given index of h_s code is called as Global probable keyword.

Local Probable keywords: The most prominent keyword identified for a given index of h_s-24 code is called as Local probable keyword.

Minimum bounding rectangle: It is the minimum rectangular boundary under which a given 2-dimentional object can be defined in computer.

Most frequent Keyword: The keyword that has the maximum frequency of occurrence at a given location

NoSQL: These are the conventional data storage and retrieval techniques followed to store non schematic data in non-tabular format.

Projected Distance: Distance taken to travel from maximum location to minimum location in any given h_s code

Sexagesimal Code: It is base-60 code used to define location in form of Degree. Hour. Minute and seconds

1.2 RESEARCH AGENDA

The conventional research in the field of Spatial Big Data has provided analyst with various opportunities to generate new and improve models to cater the need of the current times. With a number of solutions emphases on improving the efficiency of the existing solution, this research focus on the design novel spatial search model using user defined location code and hybrid indexer.

1.2.1 RESEARCH GAPS

Analysis of the SBD faces various challenges while dealing with conventional infrastructures. Hence the current research plans to provide solution to the following research gap:

- Spatial data is consider as the attribute data with associated location value. The conventional R-DBMS system generate extra column of location of numeric data type hence losing the essence of spatial data.
- Spatial dependency is the integral part of any spatial analysis. The conventional databases (SQL / NoSQL) adheres to the spatial dependency either by overlaying logical model or by creating intermediate layer [Prakhyath et.al. (2015)].
- Spatial data is stored as native data types hence multiple information get lost while enveloping the location data in such datatypes [Dong-Wan et.al. (2015)]
- Conventional architecture designed based on r-DBMS is inadequate in

catering the needs of SBD.

- All GIS system utilize UTM grid to generate location code in sexagesimal format either in Degree or in DMS value.
- During the processing of search spatial data store in existing spatial architecture, Inverted index is used solely to index textual keywords in the spatial attributes [Zhang et al. (2015)].
- Indexing techniques followed spatial infrastructure treat attribute indexer as different form location indexer [Zhang et al. (2015)].

On the bases of the above mentioned research gaps the problem statement of the current research is formulated.

1.2.2 PROBLEM STATEMENT OF RESEARCH

Today's world is changing dynamically in perspective of handling of ever increasing data volume. Various devices are generating data for the same phenomena at different time and location. GIS provides a solution in which each and every parameter is evaluated against its physical location and time of generation of data and store such information in Geodatabase data model. Contemporary "Geodatabases" uses traditional relational-dbms systems which are designed to work with predominantly structured data and utilize concepts of normalization for optimal storage. With the paradigm shift of data from structured to unstructured/semi-structured, the prevailing search techniques need to be revised. Therefore four different problems are identified in the current study. These problems are as follows:

1. Additional processing cost of indexing geographic objects are added to search algorithm [Zhang et al.(2015)]:

While dealing with geographic object both textual and spatial indexing techniques are required and hence additional implementation and maintenance cost of the search algorithm. The above identified problems ascertains the necessity for development of a new and more refined spatial database search algorithm, which would be able to handle the current requirements more reliably and promptly.

2. Agile nature of the existing data search algorithm while dealing with paradigm shift:

Present algorithm performs analysis in linear time but spatial data work in cubic time and hence unable to present desired outcome [Jardak et. al (2014)]. Recent studies suggested that column database based search algorithm can be optimized for fast retrieval of spatial objects. [Jardak et. al (2014)].

3. Effect of storing spatial data in pre-defined data types:

Spatial data has two components that are attribute and geographical location. Attribute can be stored in any standard database schema but while storing geographic location Blob or Lob datatype are used [Jardak et. al (2014)]. There is compatibility issue with some GIS software and SQL based storage system.

4. Accuracy rate of processing location for various global issues:

The processing of spatial data always demands additional exercise. One can either process additional new data or the existing old data. Both of these data showcase various challenges such as:

- New Spatial Data [Johnson et al (2016)].
  - Data Quality
  - Ownership of Data
- Old Spatial Data [Kyoon et al.(2006)].
  - Miss-registration of data
  - Lack of Metadata.
  - Non uniform quality.

1.2.3 IMPORTANCE OF THE RESEARCH

During this Pandemic, various different field of analysis has evolved to provide solutions to the existing problems. The models designed by any research oriented company thrives to visualize there results on maps that can be addressed by the masses. Therefore restating the importance of GIS and spatial analysis. The current research in the field of spatial search algorithm will benefit the society by presenting the novel spatial representing, indexing and searching techniques. Identification of spatial hotspots are the need of the hours and its application prospect in the field of Social Media Analytics, Navigation and Transportation industry, Infrastructure and Urban Regeneration sector is in high demand. The current research will also share light on the application of NoSQL based database such as Column database in providing solution for fast searching of spatial queries. Not just improving the existing infrastructures the current research will introduce a new location representational techniques which will change the course of location data processing paradigm. Not but the least the current research will support in the initiative of spatial data democratization in India.

1.3 RESEARCH HYPOTHESIS

The current research follow on the associated hypothesis where change in the query location will alter the outcome of the model. The CBDFI model which will successfully convert every location into "h_s code" and will produce reconversion strategies too. The CBDFI model will return a positive match from the provided dataset for any mentioned keyword in the spatial query. And the CBDFI model is a self-learning and self-aware which will positively improve its efficiency by reducing seek time as the distance from the query keep on increasing.

1.3.1 RESEARCH QUESTIONS

The current research purpose a novel CBDFI model to search spatial entity for any given datasets. The model create weighted tree of h_s location superset and search most probable match for the spatial query. Based on the hypothesis designed for this research the following questions will be answered:

Q1: How to identify the location of the desired entity which lie in the given distance and meet the query keywords?

Q2: How the CBDFI model will convert the location of spatial entities into "h_s codes" and calculate the probability of each "h_s code"?

Q3: How the CBDFI model will identity the most frequent most probable and most prominent keywords for each given spatial queries.

Q4: From the given search query location how the CBDFI model will generate Hs- I tree and calculate the weight of traversing among the node and identify the hs-12 node.

The proposed research will plan to answer these questions and based on the research gaps. Problem statement and research question the objectives of the research is decided.

1.4 RESEARCH OBEJCTIVES

With the technology advancement in information technology, data processing with multicore processor and the need of hour to manage the dynamically referred geometric, geographical data, it is very essential to design novel spatial database architecture with optimized data search techniques. Hence the objective of the current project is as follows:

"To design a novel search optimization algorithm for Logic based deductive Spatial Database".

The proposed research will provide one with optimized search algorithm which will overcome various problems encountered while working with spatial data in the era of Big Data. The current work will have the following outcomes:

1) To perform comparative study of the existing search algorithm
2) To design and develop novel spatial search algorithm
3) To design datatype for Column based spatial database framework
4) To implement the novel algorithm and test its efficiency with existing standard search algorithm of spatial database.

1.5 OUTLINE OF THE THESIS

The thesis is divided into the following seven chapters:

Chapter 1 Introduces the basics of Spatial Big Data with motivation to the proposed work. It is followed by the introduction to spatial data characteristics. Chapter also highlight the agenda of the current research by stating the research gaps, problem and hypothesis. This chapter present the objective of the current research.

Chapter 2 is devoted to the literature survey conducted to understand the role of various different database used to store spatial big data. Chapter also states the challenges in processing spatial big data and explain the evolution various different search indexes and search mechanism followed to process the spatial data.

Chapter 3 explain the dataset used for the current model. The chapter share light on the different data cleaning techniques follow in this research. The chapter explain the various indexing algorithms used for spatial search and explain the custom algorithm followed in this research.

Chapter 4 describes the methodology followed to achieve this research work. The also highlight the silent feature of the proposed model for searching spatial data in the given column driven approach

Chapter 5 presents a unique location code called as the "h_s code" which help in presenting the sexagesimal location into a 7 bit hex code and application to the hidden markov model to traverse among the code

Chapter 6 describes the spatial index "Hy-S" indexer which can be used for both textual and spatial indexing. Chapter also highlight the "Hs-I" tree which is a novel data structure used to store the spatial location. And finally it present the novel search algorithm.

Chapter 7 sums up the results of existing and the proposed methods. It presents a comparison of results of the Geohash and “h_s code”. It also present the comparative result of the comparison of indexer and CBDFI search model with I3 Indexer based top k spatial keyword search.

Chapter 8 concludes thesis with research directions and future scope in spatial data search followed by bibliography and list of research publications in Appendix A

CHAPTER 2

LITERATURE REVIEW

The literature review of the research work has focused on three areas. The first area identify the various different research carried out globally in the field of database storage technologies to store and process spatial data. The second part identify the limitation of the existing system and third part bring insight to the various different search approaches followed to process spatial data. This chapter identify the problem statement and list of objectives for this research.

2.1 COLUMN DATABASE FOR SPATIAL ENVIRONMENT

Traditional relational database management systems are inadequate to process large data in range of petabyte [Mohan (2013)]. In order to achieve faster and cost effective data access physical architecture of database is reformed. Cluster of inexpensive commodity servers that force distributed processing are exploited. This new transformation of database is called as NoSQL i.e. Not Only SQL database. NoSQL database not only provide distributed processing and inexpensive hardware but presented a different approach to maintain data integrity and consistency. In distinction to ACID property of RDBMS, NoSQL database exercise CAP theorem i.e. Consistency, Availability and Partition Tolerance. The downside of CAP is that of the three possible combinations, only two can be achieved at an instance [Doulkeridis et.al. (2014)]. Therefore over the time CAP has progressed into BASE (Basically Available, Soft state and Eventual consistency). Basically Available means that data will be available in spite data retrieval failed or retrieved data is inconsistent in nature. Soft state means that the data will keep on changing over time to seek consistency. And Eventual consistency means that with every update, the database will achieve consistency.

During the beginning, relation database supports notion of uncommitted reads and different levels of locking therefore accompanying BASE is not crucial issue nowadays. NoSQL column-family database supplement Schema Read feature. NoSQL database can scale data both in horizontally and vertically direction [Abramova et. al. (2014)].

2.1.1 LITERATURE SURVEY ON DIFFERENT DATABASE USED FOR SPATIAL DATA PROCESSING

The customary data processing and handling methods has been challenged by the precipitous growth in data volume [Sun et.al. (2017)]. Eskandari L. et.al. [Eskandari et.al. (2016)] paper stated that the contemporary technologies are crafting data at expeditious rate approximately 30,000 Gigabytes per second. Secondly, report published by international data corporation state that the current decade (2010-2020) is monitoring doubling of data per year [Giuliani et.al. (2014)]. Big data analytics is the best solution for working with such kind of data. Study conducted by Veda C. Storey et.al.; has thoroughly examine the characteristic of the data and stated which characteristic can be handle by which domain (technological domain or software domain) [Storey et.al. (2017)].

Table 2-1: Characteristic of Big Data and there solution

Characteristic of Big Data	Technology Solution	Software Solution
Volume	Yes	
Velocity	Yes	Yes
Variety		Yes
Veracity		Yes

Source: Singh et.al. (2019): Future Prospects and Challenges in Geospatial Database for Handling of Big Data Concept: A Review

Eric Evans distributed database studies has pointed that data driven paradigm require radical change in data storage techniques, hence use of NoSQL database are encouraged [Corbellini et.al. (2017)]. NoSQL is BASE theorem based data storage mechanism, which follow no fixed schema and provide horizontal scaling. Study conducted by Veda C. Storey et.al; has mentioned various different database and their capability to work with BASE theorem table 2-2 [Storey et.al. (2017)].

Table2-2: NoSQL vs. SQL database

Data-base	Relatio-nal	Sql-Query	Column store	Scaling	Eventual Consistency	BASE	Large Data Volume	Schema
SQL	Yes	Yes	No	Limited	-	No	NO	Fixed
NoSql	No	No	Yes	Yes(Horizontal)	Yes	Yes	Yes	Yes

Source: Singh et.al. (2019): Future Prospects and Challenges in Geospatial Database for Handling of Big Data Concept: A Review

Another study conducted by Ali Davoudian et.al. & Abdul Haseb et.al; presented classification of NoSQL Database and comparison of different database available in market table 2-3.

Table2-3: Different NoSQL database

Low: Less possibility of alteration Medium: Moderate possibility of alteration High: Great possibility of alteration

Parameter	**Data Models**	**Scalability**	**Data Size**	**Data Complexity**	**Communication Protocol**
Neo 4j	Graph	Low	Low	High	SSL
Dynamo	Key Value	High	High	Low	HTTP
CouchDB	Document	Medium	Medium	Medium	SSL
MongoDB	Document	Medium	Medium	Medium	SSL
Bigtable	Column	High	High	Low	
Hbase	Column	High	High	Low	SSH
Casandra	Column	High	High	Low	SSL
MariaDB	Column	High	High	High	SSL
MonetDB	Column	High	High	Medium	SSL
C-Store	Column	Medium	High	Low	SSL

Source: Singh et.al. (2019): Future Prospects and Challenges in Geospatial Database for Handling of Big Data Concept: A Review

Microblogging is a crucial part of everyone's life. Various studies have shown that such practices are rich source of timely data forvaluable information [Sun et. al. (2017)]. Smith et al; has stated that 88% of US adults use internet, 77 % owns smart phone and 69 use social media. 30 % of this tagged location in their post [Zhang et.al. (2014)]. Geo-tag information over microblogging sites is Volunteered Geographic Information (VGI), which is pervasive in nature, and make each citizen a sensor. Cugler et. al. paper has also mentioned the role of VGI in increasing velocity of geospatial data. These improved service models provide broad geospatial information to upkeep social popularization (Ye 2008; Luo et al., 2009).

Geographical data is consolidated in hierarchical data object by spatial database. ArcGIS is one of the most important solution provider to process geospatial data. White papers of ERSI has stated working principals of ArcGIS's geodatabase. "Working with the geodatabase: Powerful multiuser editing and sophisticated data integrity" in 2012 demonstrate the structure, working and various feature of geodatabase [Geodatabase (2012)]. Another paper "Understanding Coordinate management in geodatabase" in 2007 illuminated the multi dimensionality of geodatabase that is in 2, 3 or 4 dimension. This paper also exemplifies the importance of m dimension.

Both SQL and NoSQL based database management systems has provided solution for geospatial data. MySQL and Postgre-SQL has incorporated OGC's SFS and SFSQL to provide the functionality of spatial analysis, but face the problem with scalability of the current data. In S. Schade, (ISPRSarchives, 2015) paper optimal treatment of the big data to handle geospatial data was introduced. In order to investigate the big data of geospatial nature presented over distribute geography a different approach needs to be reformed. To deal with the large scale of unstructured data column based NoSQL database such as Neo4j, RIAK, Hbase, MapReduce and Cassandra can be used [Medina et.al. (2012)]. V .Kantere et al paper stated that in case of decentralized spatial information, a peer to peer paradigm can be beneficial [Tobler (1970)]. To achieve fast retrieval of information through query processing Online analytical processing is used. Usually OLAP operations work on relational (ROLAP) or multidimensional (MOLAP) data architectures (P.F.R. Silva et. al). Another paper by Max Cheualier .et .al, acme the preface of horizontal scaling in standard OLAP based Data warehouse [Zhang et. al (2014)].

To process spatial big data the best frame work presented is over distributed file systems like Google file [Giuliani et.al. (2014)] system and HDFS [Borthakur et.al. (2007)] which further utilize map-reduce [Dean et.al (2008)]. MongoDB and HBase has provider analyst with advancement of Column-oriented database systems which support OLAP or join processing. Zhange et.al. paper stated that Apache Hadoop is a framework that facilates parallelization, remote execution, data distribution, load balancing, or fault tolerance while working with SBD [Zhange et.al. (2014)]. Other than Apache Hadoop; Pregel [Zhang (2014)], GraphLab [Nguyen et.al (2017)], Power-Graph

[Zhang et.al.(2014)], HaLoop [Bu et.al. (2010)], PrIter [Geodatabase (2012)], and CIEL [Nguyen et.al (2017)] also provide solution for processing SBD. Lee et.al.; paper stated how Complex event processing (Oracle CEP and Esper) and Spatial-OLAP (JMap and GeoMondrian)can be used to manipulate SBD. Major CEP engine doesn't provide parallel processing but Interstage Big Data CEP Server by FUJITSU [Liu et.al. (2016)].

2.2 LITERATURE SURVEY ON LIMITAION OF EXISTING SYSTEMS FOR SPATIAL DATA HANDLING

Veda C. Storey et.al. Paper has highlighted some of the shortcoming of tradition database systems while working with such data [Storey et. al. (2017)]. These drawback are 1) single point of failure, 2) expensive with respect to amount of data, 3) impedance mismatching (aggregated verses atomic value [Sun et. al. (2017)], and 4) distributed processing (high complexity new node to data balance [Corbellini et.al. (2017)] and performance decrease as join and transaction is difficult in distributed environment).

In Grace Park et.al; paper, it states that considering the volume of the data, big data analysis may also fail due to the following error / reasons [Zhang et. al (2014)]: 1) lack of data content, 2) Inaccurate Metadata and 3) batch oriented system and their issues with real time data processing.

With increasing number of analysis proceeding at same time a new problem came into existence i.e. the problem of high concurrent access (Yang and Huang, 2013).

Cugler et. al. paper stated that MapReducd framework faces problem while working with multiple iterations.

2.3 SPATIAL SEARCH MECHANISM

The R-tree data-structure is used in majority of the search algorithm working with spatial Data. With the rising demand of the spatial data analysis a naive need aroused for the optimal search mechanism. Various researches has presented different search algorithm for spatial data depending on the application. Different search algorithm such as "Reverse spatial and textual K nearest neighborhood (RSTkNN) Algorithm" using intersection union R-Tree (IUR-tree), Clustering with Local Search CLS algorithm, Reverse Farthest neighbor (RFN) query search algorithm using NVD, and x-hop algorithm using social network aware IR-tree (SNIR-tree) for processing social aware top-k spatial keyword (SkSK) query. Most of these algorithms are loosely couples with the database and facing the problem of Euclidean distance

2.3.1 SPATIAL KEYWORD SEARCH

Spatial dimension of web, have presented researcher with wide variate of geo-location object with rich attribute description. Different POI (point of interest) available for processing are geo-coded and geo-positioned in nature hence require efficient textual and spatial indexing. In order to retrieve spatial relevant POI, spatial keyword queries are fired over real-life application such as Google Maps, Foursquare, Twitter etc. In order to retrieve most appropriate spatial object for a user, spatial query is processed on the dataset. These query return object that matched the location, keyword or both keyword and location. Based on the two deriving factors i.e. location or keywords different ranking relevance factor is calculated. Hence the conventional spatial query must return spatial object that are nearest to the query location but must be textual revenant too. Among the various spatial keyword query proposed by the research community; these three are widely popular i.e. Boolean K query, the top-k query, and the Boolean range query (table 2-4).

Table 2-4: Different type of spatial search queries.

Boolean K query	Return k object that are nearest to query location and have the same keyword in the description as mentioned in the query.
Top K query	Return k objects nearest to the query location based on the ranking score. This ranking score is combine score of spatial and textual relevance. And the return order is in ascending order of ranking score.
Boolean Range Query	Return all object who have same keyword in the text description as mentioned by the query within 10km buffer range of the query location.

Among the three mentioned queries Top K spatial keyword query is commercially proven and most of the researcher deploy the same. A top-k spatial keyword query return top k spatial object, ranked on the basis on both minimal distance from query location and relevance of object keyword with the query keyword. Top K spatial query return the matched object based on the values of k, i.e. the query will return the object with exact keyword match if the value of k is 1, if the value of k is 2 then the query will return object with better textual relevance.

2.3.2 SPATIAL INDEXER

Index is a reference pointer which help in fast accessing the record from any storage platform. In order to avoid sequential scan of every record, spatial index are generated using minimum bounded rectangle for efficient search. Hence spatial index is a data structure used to store the optimal reference of spatial object in time bound application and while working with spatial big data. For efficient working of range query, spatial join and K-Nearest neighbor, optimal spatial index is a must. Riguax et al. (2002) categorization divided spatial indexes into space driven structure and data driven structure. Mapping after partition approach is followed up in space driven structure. B+-tree extension can be mapped with such indexes hence produce indexes which are memory space and time efficient. Latter index that is data driven structures are designed on spatial containment relationship and are part R-tree family.

Some of the commonly used spatial indexes are fixed grid index, quad-tree , kd-tree, Geohash and r-tree. Among these the first four are space driven spatial indexes and the last one is data driven spatial index. As the name suggest in fixed grid index a predefined grid of fixed dimension m x m is designed with each cell is intersection or overlap of spatial objects. Multilevel grid hierarchy is used to avoid redundancy with each cell following space-filling curves. Quad- tree is widely used spatial indexing techniques in which each node is divided into four nodes and partition of the area continue until and unless the most suitable MBR is identified. Now in KD-tree axial split of the study area is executed depending of the value of the coordinates. A Geohash is a binary string each bit represent alternative division of UTM zones. The length of Geohash could range from 1 to 12. A longer Geohash has a finer precision. The second family of spatial indexes that is R-tree, form heuristic optimization of MBR to formulate spatial index that follows hierarchical order. Various version of R-tree are used globally designed to carter specific spatial problem.

2.3.3 LITERATURE SURVEY ON DIFFERENT SEARCH MECHANISM

Spatial search required to search attribute and location separately [Costes et. al (2019)] [Georgiou et. al (2019)] [Zheng et. al (2016)]. Presenting the perspective search of the information, indexes are generated both keyword index and spatial index. Based on the core design techniques, the spatial indexes can be presented as R-tree based [Zhang et. al (2016)] [Zheng et. al (2016)], Grid Base [Griffith et. al (2016)] [Giannotti et. al (2007)], and spatial filling curve based [Hong et. al (2017)] [Griffith et. al (2016)] [Zhang et. al (2016)] . Among these three, the R-tree based index presents much more efficient and moldable results, hence current paper presents the literature supporting R-tree and its applications to cater spatial search. Conventionally to engender keyword search index, inverted index or hash table are used to index keywords of large dataset [Zheng et. al (2016)]. For spatial indices include inverted R-tree [Zhou et. al (2005)] , SFC-QUAD [Christoforaki et. al (2011)], S2I [Zhang et. al (2013)] , IR2-tree [Felipe et. al (2008)] , KR∗-tree [Zhou et. al (2005)] , IR-tree [Wu et. al (2011)] [Cong et. al (2009)] , WIBR-tree [Wu et. al (2011)] , and SKI [Cary et. al.(2010)] . [Zhou et. al (2005)] paper, generate keyword-object list and create R-tree for each keyword in search to form IR-tree. With the arrival of the query only the R-tree supporting all the keywords are addressed for which incremented nearest neighborhood technique is used [Griffith et. al (2016)]. [Christoforaki et. al (2011)] paper, fused mathematical tool such as space filling curve with a keyword file generated using the inverted indexes. To reduce processing time, keywords were blocked together and the parent index of the block is searched in the S2I [Rocha et. al (2011)]. Spatial similarly to S2I tree, IR2-tree incorporated unique reference file with each leaf node of R-tree [Felipe et. al (2008)]. Li et al. proposed BR-tree [Griffith et. al (2016)], where R-tree location of the objects and B-Tree organize keywords. The algorithm followed two approaches that is keyword search first or location search first. [Griffith et. al (2016)] [Zheng et. al (2016)], paper combine both IR-tree and IR2-tree to formulate IL- Quadtree. In IL- Quadtree, for each keyword linear quad trees is generated using Morton code and different bits are used to present the status of quadrant [Griffith et. al (2016)].

For searching for confined area the KR∗-tree was proposed by [Hariharan et. al (2007)], IR2-tree is the other variant of IR-tree proposed by Cong et al. [Li et. al (2012)] [Li et. al (2011)] [Cong et. al (2009)]. A Similar IRLi-tree store Integrated Inverted file at the nodes. Other remarkable research combining R-tree with inverted index are DIR-tree, CDIR-tree, introduced by Cong et al. and SKI proposed by [Cong et. al (2009)]. For supporting multi query search, WIBR-tree was proposed by [Wu et. al (2011)]. With every tree structure proposed the specified query was solved but effect the MBR and result in change in processing time.

Top K spatial keyword search query which follow scoring function or kNN (k nearest neighborhood) to find the relevant results formulate the base for various model to find stationary and non-stationery (in-motion) objects. Various spatial search queries are designed for stationary object [Zheng et. al (2016)] [Christoforaki et. al (2011)] [Felipe et. al (2008)], where one find the probability of having an object at particular location depending upon the list of keywords given as stated above. For mobile objects, a location-aware top-k text retrieval (LkT) [Cong et. al (2009)] query was proposed by Cong et al. paper; along with query engine for bi directional motion in as MkSK query and RSTkNN query [Lu et. al (2011)] . These query engine use language models and a probabilistic ranking function to find the best fit for the location aware query. Another study carried out by Chen et al. presented Boolean range continuous query and proposed IQ-tree to support logically related keywords. [Zheng et. al (2016)] proposed I3 Integrated Inverted Index to support logical join among the search query. Different researcher presented Top K with different input criteria's such as sliding window [Griffith et. al (2016)] , distributed database [Zhang et. al (2016)] and adjusting weights of the scoring function dynamically [Hong et. al (2017)].

[Cao et. al (2010)] paper propose a model that find out result based on the priority of the keyword used in location-aware top-k prestige-based text retrieval (LkPT) query. Literature presented by Cho. X et al. supports extracting block of output object, m closest keywords (mCK) query and collective spatial keyword querying (CSKQ) [Cao et al (2011)] return all object holding minimum distance from the object suggested by the keywords and minimum distance between the output objects.

Different research carried out by [Alvares et. al (2007)] [Yan et. al (2011)] presented work to capture semantic search pattern of the path travelled. All of these research also used the top k spatial keyword search query with sematic engine. Literature suggest that Top K spatial keyword search query can be used for various other perspectives such as navigation [Rocha et. al (2011)], location based type search [Wu et. al (2011)], and rigorously ranking of an event, based on location and region of occurrence [Zheng et. al (2016)]. Conclusive from the related work, Top K spatial keyword search query is most optimal technique to deal with rapidly generated spatial data over various platform; prominently social media.

2.4 INFERENCE

From the various literature studied in the field of the spatial database, conventional systems designed for catering spatial big data and spatial search mechanism, it can be concluded that spatial big data has huge potential for research. One can generate advance architectural infrastructure with evolving technologies in the field of SBD. The literature identified in this research easily support the objectives of the proposed model.

CHAPTER 3

BASE ALGORITHMS FOR CBDFI MODEL

The proposed model is tested on an airplane dataset, provided by the aviation department of the USA. The dataset is provided by Federal Aviation Administration FAA report on the forecasting of airplane activity are various different airports under its jurisdiction. The dataset provide detail about the aircraft, there travel history, and there monthly delay status and cause. This dataset provide information regarding capacity of aircraft, duration of travel with source-destination station, staff information, passenger information with luggage report, passenger reviews and aircraft maintenance report etc. The organization planned to prepare airplane planning forecast based on the pervious years forecast called as base year and present for upcoming year in general the consider last decade report to forecast for upcoming 15 years. There are various different resources from which this dataset can be acquired such as www.apa.data.faa.gov, www.webcentral.bts.gov/oai/sources, and www.faaa.gov/arp/pdf/v3a.pdf etc. The current studied intend to find out various different location where a particular aircraft encounter delay due to same reason of faulty machinery. The model will be trained to identify the location of the nearby aircraft form the same group of airplane and the Docker station of the aircraft in repair. This dataset is freely available in text (tab delimited) format for research purpose and provide year wise finding of 2500 flights departed from New York and Washington DC to South America, Africa, Europe, United Kingdom and India.

3.1 DATA CLEANING TECHNIQUES

Data capture for the current research has to undergo series of process to ensure the performance of the search algorithm. The pre-processing of the dataset include 3 main steps of identification of error, detection of error and correction of errors [Ridzun et.al, 2019]. The basic characteristic of SBD is hetro-types data hence in the process of data cleaning remove error data, ensure consistence of data and transform the data into standard format [Ridzun et.al, 2019-7]. In the current research the data cleaning of the aviation data is carried out in the following steps:

Step1: Transforming tab delimited into schema full dataset. While transforming the data it is ensured that every column generated should have the proper heading hence if the heading is missing the appropriate header is allocated to the dataset. And secondly the location provided by the FAA is in meters i.e. projected projection system. This location need to be converted into sexagesimal format hence the purpose model take sexagesimal location format as input.

Step2: Data inspection is carried out to identify errors and inconsistency in the transformed dataset. Two different approaches are carried out in the current research to identify anomalies. To identify the error in delay related data set; data profiling is carried out and to identify the anomalies is heuristic study of aircraft prediction concepts of data marts will be performed.

Step3: In this step the transformation or the error-some data will be carried out and replace in the dataset to ensure the performance of the proposed model.

Ideally there are four type of data cleaning techniques followed for SBD, these model are Cleanix [Wang, et al. (2014)], SCARE [Yakout, et al. (2013)], KATARA [Chu et.al (2015)] and BigDansing [Khayyat, et al. (2015)]. Among the four different model the proposed model use KATARA technique as this data cleaning technique perform sematic table interpretation, this tool provide an mean to generate possible correct and wrong data and most importantly it run of the functionality of top-k result similar to the approach this proposed model present. The detail working of the process is explained in chapter 6.

3.2 INDEXING ALGORITHM FOR SPATIAL BIG DATA

Spatial data are nominal attribute data, accompanied by logical layered view over it. Indexing of the data set in fetching query plays a very important role in calculating the execution time of query. Various research have been conducted to produce most efficient indexing as per the application. In CBDFI model, a hybrid indexing techniques is used with combine inverted index, hash table and quad tree together to produce fast and new index for spatial search.

3.2.1 INVERTED INDEX

Different words, keyword and joining phrases form any documents but if a data structure can invert this document in such a way that all the relevant keywords and their respective occurrence can be recorded then such data structure is called “inverted index”. An inverted index consists of a list of all unique words that appear in any document, for each word, a mapping is done from content, such as words or numbers, to its locations in a database file, or in a document or a set of documents which contain that particular word appears. The sole purpose of inverted index is to allow fast full text searches, at a cost of increased processing when a document is added to the database. It is the most popular data structure used for document retrieval systems, which are used on a large scale for example in search engines. Dictionary and Posting are the two important part in any indexing system. Dictionary is vocabulary, lexicon of the various different term used in the document and posting is the position of that lexicon in the document. This same approach will be followed when indexing the h_s code too. The all the similar h_s code with be part of the dictionary and their position with the number of time that h_s code are repeated in the given keyword. During the search call the system will first check the keyword in the dictionary and from their it will be pointed toward the parent document and it position in that document. Similarly when a location will be searched the system is check the h_s code in the dictionary and then they will be pointed to the various nodes of the location quad tree where that keyword could be mapped. Depending upon the occurrence pattern that need to be followed in the search, two different type of frequency are generated i.e. term frequency and the document frequency. Term frequency refer to the number of term occurrences in a single document and document frequency is the number of documents in where the occurrence of term is registered.

In the process of Inverted index construction the following steps are undertaken: 1) Collection of the document to be index, 2) perform tokenization of the keyword to be stored in the dictionary, 3) pre-process in the token and last step is 4) For each term index the documents in which the term appears.

<u>Tokenization</u>

Let‘s assume that there is a collection of documents one wants to index. Tokenization is the process of separating pieces of a text – the tokens from each document, usually by white-space or other delimiter while omitting punctuation. While working with the h_s code the following step of tokenization is not required because the h_s code are token by itself.

Stop words

Stop words are extremely common words that appear in the collection but play minor role in the search. For instance the, which is the most common word in English text, but only rarely helps to retrieve the relevant set of documents for the user. Removing stop words from indexing process helps to reduce the size of the final index.

Normalization

“Token normalization is the process of canonicalizing tokens so that matches occur despite superficial differences in the character sequences of the tokens.”. There are two different way to normalize the tokens either by creating equivalence classes or by relating tokens by some relation.

Stemming

Stemming is a process of transforming words to its morphological root (stem). For example, words retrieve, retrieves and retrieval would be normally indexed as separated terms, but user can be interested in all of words from the set. Thus, the words are stemmed to a common form.

The inverted index yield by the CBDFI model is as following:

Keyword1 → doc1.name(occurrences no.) → doc2.name(occurrences

no.) Keyword2 → doc1.name(occurrences no.) → doc2.name(occurrences

no.) And so on

3.2.2 HASHING

Hashing is known to be a specific case of Indexing. Basically Indexing is used to partition the dataset based on a value of a field or a combination of fields. As well as it can also partition the data set based on a value of a function, called hash function, computed from the data in a field or a combination of fields. In this specific case, indexing is called data hashing. For a huge database, it is almost next to impossible to search on all the index values and through all its level and then reach the destination data block to retrieve the desired data. Therefore hashing is an effective technique to calculate the direct location of a data record on the disk without using index structure. For hashing the words in the file is encoded into compressed into fixed length values of a hash key. Hence the number of words to be scanned is reduced into smaller number and fast retrieval is achieved. The primary components of hashing are hash table, hash function, collision and collision resolution. The hash table can be storage in memory or on disk. Keyword and size of hash table plays a very important role in the efficient run of hash table. The values stored in the hash table can be a string or an integer. As integer are easy to be used as reference hence preconditioning is applies on the string where the string is converted into ASCII value. A similar process of preconditioning was carried out while calculating the average weight of the location number. With the help of hash function the key is transformed into a number which is stored in the hash table. This number is further used as an index to position the desired location of the searched word. Hashing provide fast insertion but for searching a moderation need to be maintained between the function and number of keys in hash table. While storing the keys sometime two keys get allocated for the same location and hence this condition is called as collision and process of finding alternative location is called Collison removal. A collision resolution strategy guarantee future key lookup operations that from know the query returns to the correct respective records and the problem of finding the same record on one location is solved.

Hash function is a function which is applied on a key by which it produces an integer, which can be used as an address of hash table. Hence one can use the same hash function for accessing the data from the hash table. In this the integer returned by the hash function is called hash key. There are various types of hash function which are used to place the data in a hash table such as division method, mid square method and digit folding method. In the division method the remainder decide the index of the record value and the dividend is the size of the hash table for example if the record 62,78,89,34 is to be placed in a hash table and let us take the table size is 20. Then index of the each number will be 2 for 62, 18 for 78 9 for 89 and 14 for 34. In the second method, the square of key and then the mid digit is considered for indexing. For example if we want to place a record of 3101 and the size of table is 1000. So 3101*3101=9616201 i.e. h (3101) = 162.

And in the third method the key is divided into separate parts and by using some simple operations these parts are combined to produce a hash key. For example: consider a record of 12465512 then it will be divided into parts i.e. 124, 655, 12. After dividing the parts combine these parts by adding it. h(key)=124+655+12 =791

In order to avoid collision the following methods are considered; chaining, linear probing, quadratic probing and double hashing. Chaining is a method in which additional field with data i.e. chain is introduced. A chain is maintained at the home bucket. In this when a collision occurs then a linked list is maintained for colliding data. While in liner probing is very easy and simple method to resolve or to handle the collision. In this collision can be solved by placing the second record linearly down, whenever the empty place is found. In this method there is a problem of clustering which means at some place block of a data is formed in a hash table. If one is working with complex keywords and confined hash table then quadratic probing is used where the hash function is defined by the H(key)=(H(key)+x*x)%table size. Double hashing is a technique in which two hash function are used when there is an occurrence of collision. In this method 1 hash function is simple as same as division method. But for the second hash function there are two important rules which are

1. It must never evaluate to zero.
2. Must sure about the buckets, that they are probed.

The hash functions for this technique are:

H1(key)=key % table size

H2(key)=P-(key mod P)

Where, **p** is a prime number which should be taken smaller than the size of a hash table. The hash function used in the CBDFI model is as follows:

Hash (key) = (sum of ASCII values of First, Middle and last character of the a word) % n

Where n is a prime factor taken from p{alpha} set.

3.2.3 QUAD-TREE

A Quad-tree is among the most common data structured used for performing spatial indexing. Quad-tree is bounded by limit for each node and in the case of outreach of the limit the node is subdivided into child node and these process keep on repeating until and unless the required child element is within the desired limit. The limit for the current research quad-tree is based on the calculated weight of the "h_s code". The first "h_s code" and its respective weight is consider as the parent node and on arrival of next "h_s code" the weights and probability of reaching that node is compared. If the weight of the "h_s code" is more than the parent node then interchanging of the nodes will take place. Any keywords k_i inside a file will be associated with the weight of the inverted file dictionary and hence there could be only limited keyword that can be written into the file location and in the memory. Hence the relation of keywords to the size of the file/memory decide the limit for the tree cell. With respect to the earth surface, the whole earth is divided into four zone i.e. east, west, north, south and they can be further divided into child quadrants. The hybrid quad-tree will be proved advantageous as each keyword will be associated with a "h_s code" which will be cluster of 24 locations situated at different UTM zones; therefore with every first matched keyword in the quad-tree will optimized the search by level of 24. The presented hex code are unique for a group of 24 distinct location and will be represented as one quad-tree node. The number of the group member could be less than 24 under controlled condition. The number 24 is the result of 4C4 combination, i.e. a sexagesimal location will be having four parts for example $24^o.34'46.54''$ which can be generalized as $A^o.B'C.D''$; there are going to be 24 different combination of the ABCD which will generate the same h_s hexadecimal keyword with values of A, B, C, and D less than 60. But if the value is more than 60 then the combination values will be less than 24. This group of 24 location present us with a means to check more than one location every time someone is investigating any "h_s code". The quad tree representation of the "h_s code" is shown in the figure 3-1 and follow a specific design to represent the locations.

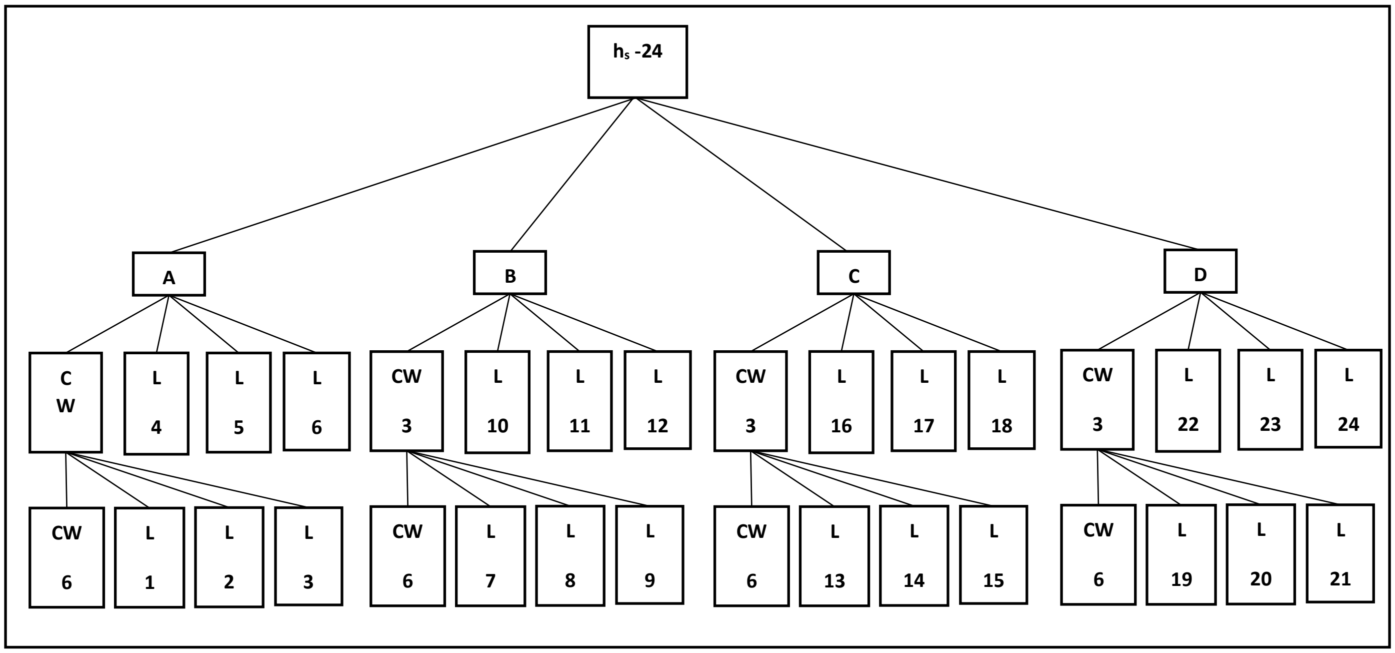


Fig3-1: hs -24 Quad tree

The left most leaf node of every child node will point out at the exact location traversal weight to identify a particular behavior of the object in study. Therefore the summarized location h_s quad tree will present the researcher with a data structure that can be used to search multiple keywords in the query being addressed by the user. The second left most leaf node of each child node will provide with the pointer to the path to the most suitable location as per the requirement of the user. If the this node is empty then one can move to the next child node quadrant as none of the present six location meet the requirement of the user. Hence while traversing through these node one can easy identify the most suitable location that can be accessed under the given circumstance. Generation of the hybrid quad-tree for the presented research will follow the following steps:

Algorithm 3-1: Generating Quad-tree from hs-24 Code

```
Step 1: Start
Step 2: Enter the I^th location value
Step 3: Generate hs code of the entered location
Step 4: Calculate the weight of the hs code generated and consider it as parent node
Step 5: Entered the next I +1^th location and generate its hs code
Step 6: Calculate weight of the I +1th hs code
Step 7: Compare the weight of I +1th hs code with parent node
Step 8: If the weight of I +1th hs code is more than the weight of parent node then interchange parent and child node else make I +1th hs code as child node
Step 9: If weight of the I +1th hs code is less than the other child node then child node with the nearest match will be further divided
Step 10: Calculate the probability of traversing from child node to parent node and store with weight of child node.
Step 11: Go back to step 4 and repeat the process.
Step 12: Stop
```

Hence the generated hybrid Quad-tree will suffice the purpose of the CBDFI model in improving the search efficiency of the SBD.

3.3 CBDFI MODEL

CBDFI model is designed to improve the search efficiency of the spatial dataset. Substantial research has been carried out in the field of SBD. Adhering to the paradigm shift from SQL based databases to NoSQL databases hybrid architecture is produced as conventional solutions to cater the need of SBD. The proposed research model intend to produce a novel search algorithm designed to work with SBD in Column database. The current model has modified the way the location is being stored and processed in any conventional product. The introduction of the hs code will provide analysts with a new domain for research and CBDFI model is once such positive experimentation. The proposed model follow the following steps:

Algorithm 3-2: Design CBDFI model

```
Step 1: Start
Step 2: Import spatial dataset from the source
Step 3: Run the pre-processing procedure to clean and parse the unstructured data set into schema enable format
Step 4: From the processed dataset generate the library of location and associated keywords
Step 5: Convert the location into hs codes
Step 6: Generate the hidden markov model to calculate the probability of the hs code
Step 7: Generate the hybrid quad tree from location's hs code
Step 8: Generate the Hy-S indexer for hs code and keywords
Step 9: Populate the Pre-fetch table with most probable keywords and their respective hs codes
Step 10: Generate the Hs-I tree with calculated weights, cumulate weight of hs-24 nodes and probability
to traverse
Step 11: Input spatial query with keyword, location and k
Step 12: Calculate the spatial and textual relevance
Step 13: Run the CBDFI search algorithm
Step 14: Render the changes in the Hs-I tree
Step 15: Return the result of the spatial query with identified locations.
Step 16: Stop.
```

3.4 INFERENCE

This chapter presented the experimental dataset undertaken to prove the working of the proposed research. The algorithms stated in this chapter will proved insight while designing the methodology for the given research in the upcoming chapter 4. The working of the CBDFI model will be explained by various different algorithms stated in chapter 6.

CHAPTER 4

METHODOLOGY OF RESEARCH

This thesis proposes a novel search algorithm to search spatial data and it utilize the unique “h_s code” representation of the location. Another major contributing is the purpose of the indexing techniques that can be used both for the textual index as well as for the spatial indexing

4.1 METHODOLOGY

In the presented work, True experimental research design with bottom up approach is exploited. In order to achieve the desired objectives, agile model will be used. The current research work will be carried out in four stages i.e. data pre-processing, Location encoding, column database design and spatial search (figure 4-1).

In stage 1 spatial data will be collected from various sources and understand its intrinsic nature. With the paradigm shift of structured data to unstructured data, Spatial Big Data would be generated. This stage will provide us with various challenges and difficulties faced while working with spatial big data. And pre-processing of the spatial data will be carried out in this stage which will be processed for designed spatial search algorithm

In stage 2 location data will be extracted and will be encoded to generate new spatial indexer for fast spatial search.

In stage 3 new data type will be developed [Barewar et.al. (2014)][Radhidan et. al.(2015)] which will overcome the problems faced by spatial big data. One of the major issues faced with spatial data is storing the spatial information in its same nature. Various other problems such as interoperability, open standard and cross platform also poses a huge challenge for developers. Hence a new data type will be introduced and incorporated in NoSQL database

In the last stage different existing spatial query algorithm will be studied and understand. Among the available the selected algorithm will be incorporated with NoSql database. Along with it various spatial analyses will be conducted to study the performance speed of data sample stored in new data type and will list down the different problem faced. The process flow of the research is illustrated underneath.

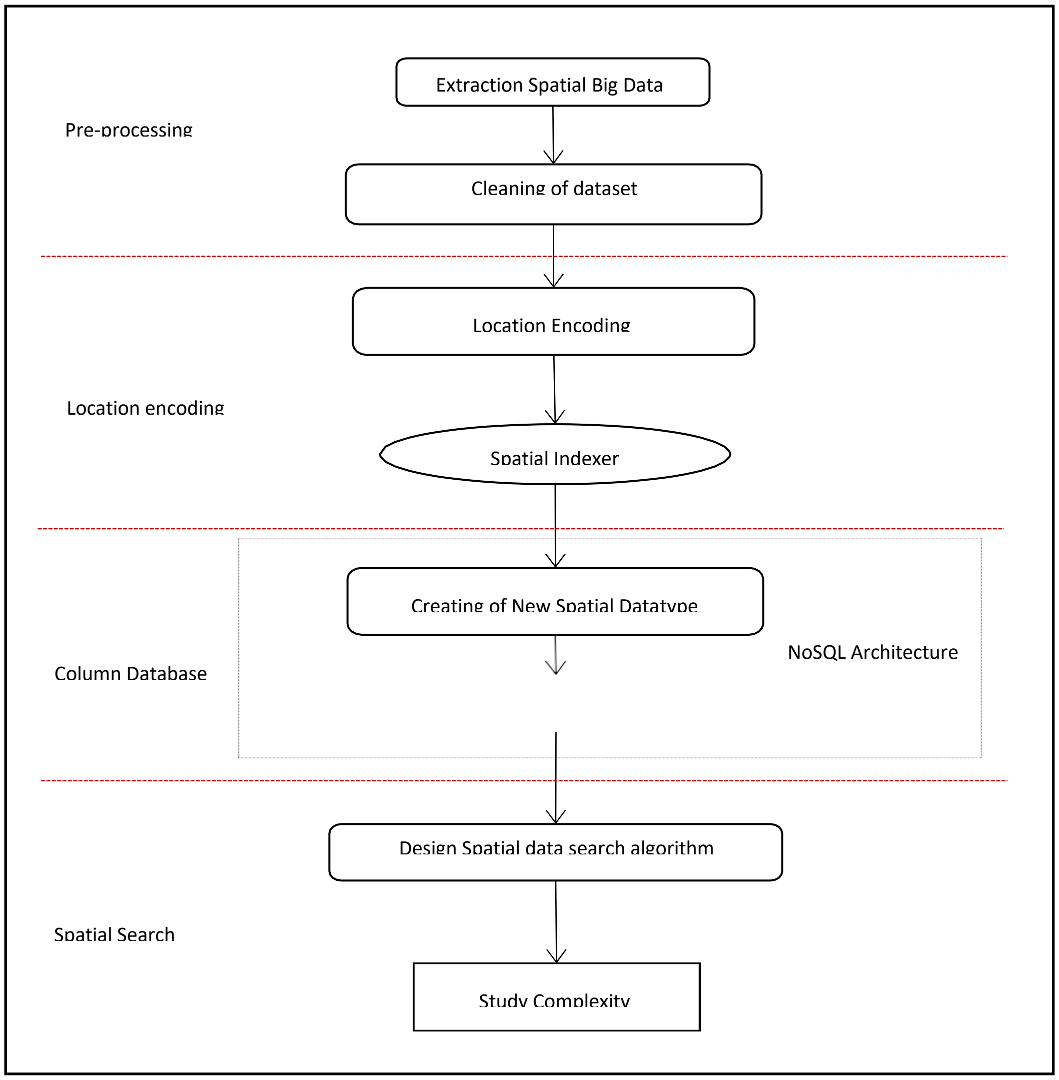


Fig4-1: Methodology design diagram

The Methodology followed for the current work is as follows:

1. Comparison of Spatial Architecture and spatial search algorithm.
2. Analysis existing data type for storing spatial data and devising new user defined data type.
3. Study the architecture of Document based NoSQL database
4. Design a logical structure of spatial architecture following BASE theorem with incorporated spatial datatype (SDT)
5. Study of different dependent parameters of spatial search algorithm and design novel

search algorithm.

6. Compare and optimize the performance of search algorithm.

Table 4-1 Methodology and outcome mapping

S. No	Steps	Desired Output	Challenges	Outcome
1	Comparison of Spatial Architecture and spatial search algorithm	Working with Spatial Big Data	Multitude of Spatial Data format, and working with Euclidean distance	Compare the existing search algorithm
2	Analysis existing data type for storing spatial data and devising new user defined data type .	New basic Spatial datatype with implicit behavior	Storing hierarchy and working with Euclidean distance	Develop spatial data type
3	Study the architecture of column oriented NoSQL database	Advantage of Base theorem	Meeting the demand of the current analysis with column based database	Design column based spatial database architecture.
4	Design a logical structure of spatial architecture following BASE theorem with incorporated spatial datatype (SDT)	Optimal searching of Spatial Data in proposed database architecture	Sustaining the intrinsic characteristic of spatial data i.e. hierarchy, actual shape and relationship	Design Logic based deductive spatial database architecture.
5	Study of different dependent parameters of spatial search algorithm and design novel search algorithm .	Parameters that can alter the efficiency of search algorithm	Incorporating the proposed algorithm as intrinsic property of the spatial database architecture	Implement the novel algorithm
6	Compare and optimize the performance of search algorithm	Processing speed of search algorithm	Improving the efficiency and reducing operational trade-off	Test efficiency of algorithm with existing search algorithm

4.2 CBDFI FRAMEWORK

Spatial data are nominal attribute data, accompanied by logical layered view over it. This logical view is responsible for the latitude/longitude of the data along with their intrinsic relational behavior. All the object exist in the reality have strong sense of dependency among each other, which define the object existence. Design of the logical view of the spatial data requires through understanding of the data along with well- defined structure, which present with the guard condition of the logical expression. Coordinate system such as Universal Transverse Mercator (UTM) is on such logical expression to design the logical view of any spatial data available for analysis. Design of the over-layer view helps in the retrieving the spatial information and perform various operations of search, insertion and updating. The complexity of the design of the logical expression increase when the analyst processes schema-less data. For such data, there are many ways to form logical expression, this paper present the "Column BASE- Data Fetch and Index" (CBDFI) model for performing a fast spatial data search. The model also used a unique location code to traverse globally to and from a certain point in reality. This presented hexadecimal code formulate the base for searching spatial data in CBDFI model.

The black-box view of the architecture of CBDFI model include data encoder, followed by spatial pre-fetch and Column database. Data encoder; extract latitude, longitude from presented structural, semi-structural & un-structural data and store then in controlled fashion for further processing. Following which sexagesimal location is rehabilitated into hybrid ASCII code called as Hybrid Spatial ASCII (h_s code) of 12 bits. The code is converted for 0 - 90o latitude; 0 - 60 minutes; 0 - 60 seconds 0- 180o longitude. The h_s code comprises of alphabet A-Z and decimal number 0-9. These codes present the similar characteristic of hexadecimal code hence h_s code is also called as hexadecimal spatial code. The generated h_s code will be stored in spatial pre fetch. The column database will provide storage schema for the whole process. The proposed framework is designed to create unique indexing key which will incorporate the h_s code and attribute indexer to formulate hybrid indexer for CBDFI model.

Hybrid spatial indexer (Hy-S indexer) ; identify the unique keyword for the each generated hs code and index these hexadecimal code with the help up of scaled up indexing algorithm Hy-S indexer combining IR-tree and Hs I-tree (hybrid h_s code index tree) which combine linear quad tree with inverted index and hashing [Costes et. al (2019)] . At the final stage of the "Hy-S indexer", the keyword and the associated index will populate the leaf node of the tree data structure. During the search call the most appropriate node and the corresponding child nodes will be picked for understanding the correlation of the keyword with the corresponding "h_s code".

The "h_s code" can associate with any keyword based on the similarities of the "h_s code". Ideally, there can be four different cases to define the relationship between the generated "h_s code" and keywords. These cases are as follows:

Case1: Each generated "h_s code" generate unique keyword. Therefor number of keyword and generate "h_s codes" are the same,

Case 2: Multiple generate "h_s codes" are associated with same keyword. Therefor better and fast indexing,

Case 3: Multiple keywords are associated with same generated "h_s code".
Therefore creating duplicate keywords and increasing the complexity and

Case 4: Multiple keywords are associated with multiple generate "h_s code".

In the CBDFI model the multiple generated h_s code are associated with single keyword, hence reducing the conversion time of the indexes and enhance spatial search. The "Hy-S" indexer will be following the standard topological vector model such that following relationship is maintain (Table 4-2).

Table4-2: Topological structure to store hs code of Hy-S indexer

Hs code	**Latitude**	**Longitude**
GCGCICO33	0"	
ICOCOCO35		18"
So on ….		

4-2. a: hs table

Index	Keyword	Lat/Long
Index1	Keyword1	0"/ 12"
	Keyword2	8"/18"
So on…..		

4-2. b: Index table

Index	Keyword	Hs code Start	Hs code End
Index 1	Keyword 1	GCGCICO33	ICICICO33
	Keyword 2		

4-2.c: Hy-S Indexer table

The CBDFI model will convert these columns, with each column stating the latitude, or longitude or altitude of an index and will return the generated “h_s code”. The columnar reference will generate the “Hs-I tree” and will facilitate the spatial search. The framework of the CBDFI model (Figure4-2) will be divided into four major areas; the data encoder, hybrid spatial query engine followed by query optimizer and finally the columnar database Monet DB. Formation of the hybrid indexer will be deriving the whole search algorithm. The created “h_s code” will formulate a linear quad tree and will support the Top k spatial keyword search query. After formation of the indexes, probabilistic model [Singh et. al (2019)] will be used to ensure the conversion of the query. For each generated index all the related keywords, theirs associated objects/entities and correlated surrounding objects will be stored. Considering the storage schema various different NoSQL databases can be deployed.

With due understanding of the dynamics of this search mechanism, column database serve the purpose [Costes et. al (2019)]. [Singh et. al (2019)] paper present a clear comparison of the different columnar databases for storing spatial data in No SQL schema. [Srivastava et. al (2012)] paper state various problems faced while integrated such data in No SQL schema. Therefore, the results will be stored in the column database and on every query fired by the user, engine will check the node of the tree for the suitable match and return the result of the query.

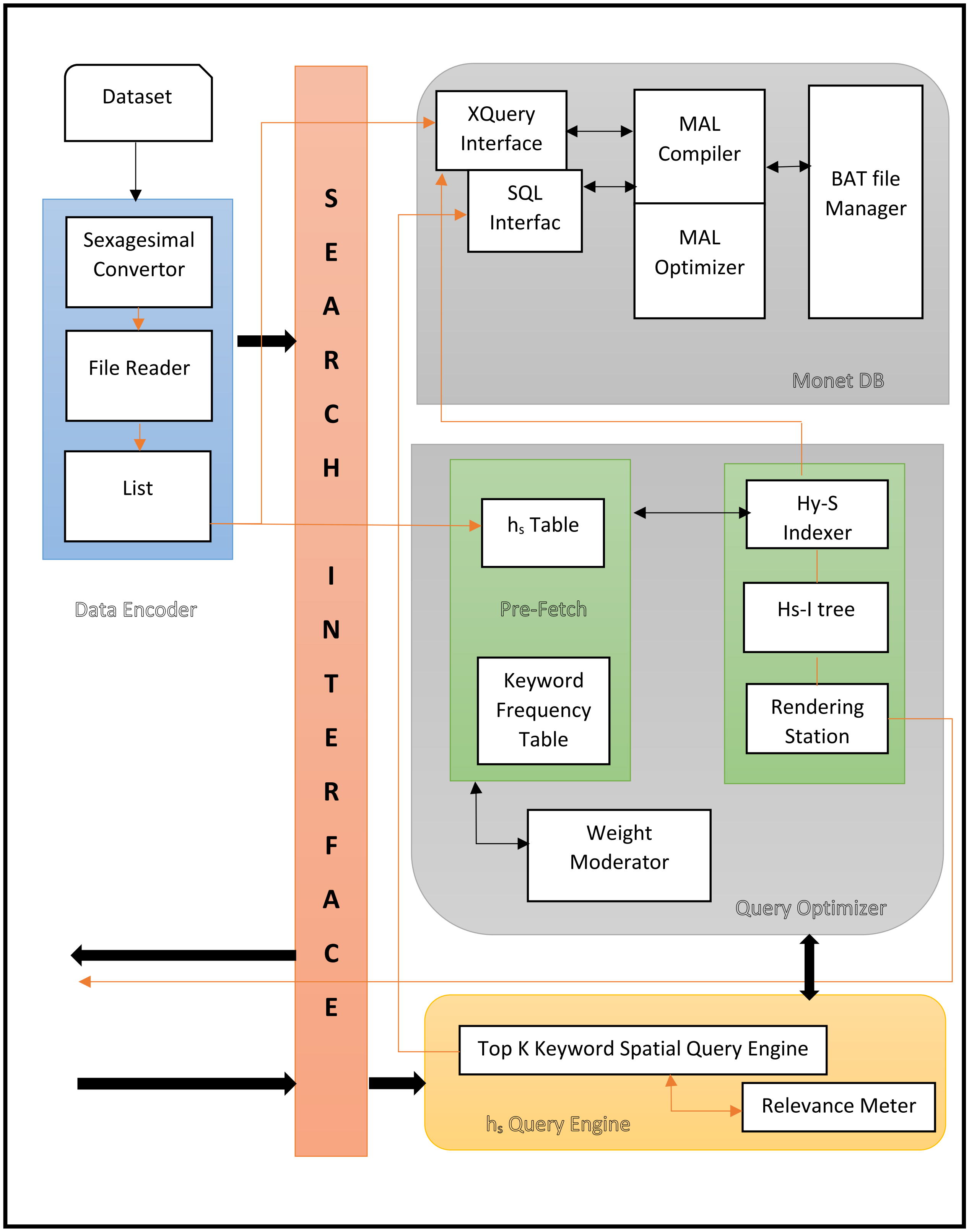


Fig 4-2: CBDFI Block Architecture

4.3 INFERENCE

This chapter present the various steps undertaken to formulate the methodology of the current research. Finding from the literature review of chapter 2 and steps formulated in chapter 3 provide the base structure for the methodology of CBDFI model. On the bases of the presented methodology the CBDFI model will be explained in upcoming chapters.

CHAPTER 5
HS CODE- UNIQUE LOCATION DATATYPE

The representation and the storage of the location code is the major reason for complicating the process of spatial search. Hence to produce a novel search algorithm a unique and refresh encoder is required to register the location and store the intrinsic behavior of the location to. With the generating of "h_s code" a new mean for processing location is introduced in this research.

5.1 HS ENCODER

The CBDFI model work on "h_s code" converted from the latitude and longitude sexagesimal value. The converted "h_s code" showcase a unique pair of character values, arranged in a manner to generate unique codes for each spatial location values. Transition of each code form a pair to another formulate the rules for searching the location values during query call. The frequency of repetition of the each bit result in forming the guard condition for search.

Prime number and prime factors plays a very important role in enhance the processing time of different mathematical calculation [Hadraba]. The h_s convertor also rely on the prime factor of the total keyword identified in the dataset. Keyword (h_s code) generator start with conversion of the extracted latitude and longitude into weighted ASCII code. As the latitude and longitude follow the standard decimal numbers system, therefor the weighted ASCII code follow the same convention. Weighted ASCII code of number are as follow

0 -99 = ASCII code of tenth place * 10 + ASCII code of unit place **(5.1)**

100- 180 = ASCII code of hundred place * 100 + ASCII code of tenth place * 10 + ASCII code of unit place **(5.2)**

For any number = ASCII code * Weight of Hundred + ASCII code * Weight of tenth + ASCII code * Weight of unit. **(5.3)**

After converting the weighted ASCII code, formation of alphabet set (Palpha) take place ensuring only alphabet whose ASCII code is prime number and these prime number is prime factor of the total keywords. The whole process of generating "h_s code" depend upon the division of weighted ASCII code from each element of Palpha{} set until and unless one finds the first prime dividend. On identification of the first prime dividend, ASCII code of the matched element p € Palpha{} is subtracted from the weighted ASCII code to find revised weighted ASCII code. The process keep on repeating until the remainder is less than 65. For example latitude 2o (see Table 5-1):

Weighted ASCII code = 48*10 +50 = 530, (from **5.1,5.2,5.3**)

Palpha {C,G,I,O,S,Y}.

Weighted ASCII code divided by Palpha : 530 / p € Palpha{} (**5.4**)

The convertor will generate the h_s code for each location in degree, minute and second format. The h_s code generated will be 9 bit long for values 0-59,63 and 10 bits for 60-62, 64-90. These 9/10 bit code will be enveloped in 12 bit format by suffixing the 3/2 bits at the MSB. The 9 bit code from 0-47 will be prefixed with Y and followed 48 -59,63 will be prefixed with I at the 9th bit.

Table5-1: Step by step conversion of Location 2° N latitude into h_s code

Weighted ASCII	First Prime Remainder	Remainder ASCII	Code
530	71	459	G
459	67	392	C
392	73	319	I
319	67	252	C
252	73	179	I
179	67	112	C
112	79	33	O

The 10th and 11th bit will comprises of 2 bit code to understand the position of the number pair in the location. The normalized code will be 12 bit which will uniquely identify the value. These 10 bit code are further converted into 8 bit code for more simplified calculation.

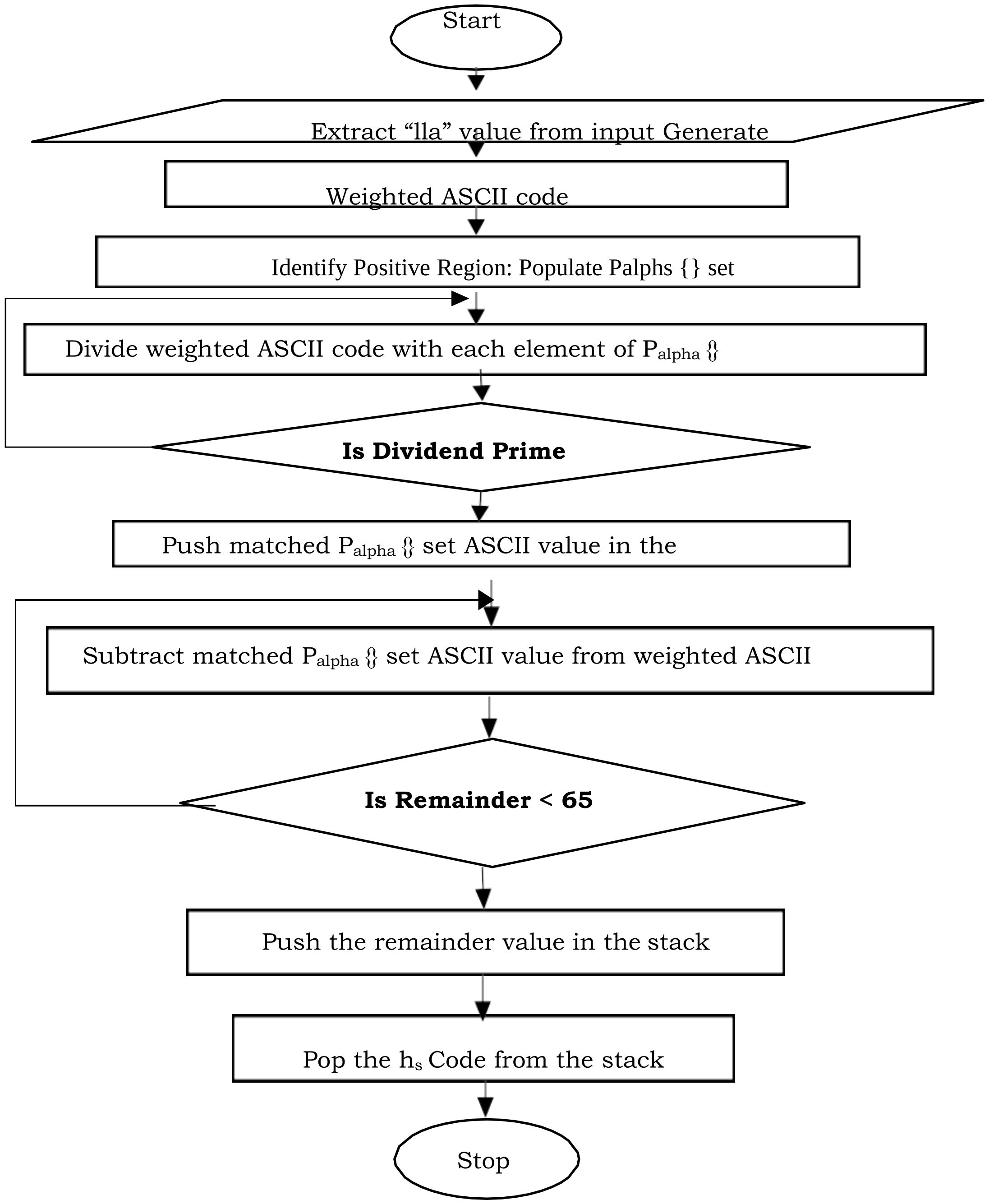
The algorithm demonstrate the working of h_s code generator

Algorithm5-1: Generating h_s code

```
1: for i in range(n,n+1):
2:         u ←   i%10
3:         u  ←  dict1[u]
4:          t   ←  int(i/10)
5:                   t  ← dict1[t]*10
6:          s   ← u+t
7:         num.append(s)
8:         print(num)
9: for i in range(len(num)):
10:        while j<7:
11:                  while(num[i]>dict2[j]):
12:                    temp ←   (num[i] / dict2[j])
13:                      temp  ←  round(temp)
14:             if(temp - int(temp) <= 0.54):
15:                   temp  ←  int(temp)
16:            else:
17:                  temp  ← round(temp)'''
18:                  if(sympy.isprime(temp) == True or temp==1 or temp==0):
19:                  z  ←  dict2[j]
20:                  k  ← chr(z)
21:                        num[i]-   ← dict2[j]
22:                        key   ← key+k
23:                  j   ← 0
24:                  else:
25:                            j   ←  j+1
26:                  break
27:         if(num[i]<dict2[j]):
28:                             break
29:                  if(num[i]<67):
30:                            key +      ←  str(num[i])
31:  result.append(key)
```

```
32:  key   ←      ""
33:  print(result)
```

The following process flow converting the location value into h_s code is presented in Figure5-1



Start
Extract "lla" value from input Generate
Weighted ASCII code
Identify Positive Region: Populate Palphs {} set
Divide weighted ASCII code with each element of Palpha {}
Is Dividend Prime
Push matched Palpha {} set ASCII value in the
Subtract matched Palpha {} set ASCII value from weighted ASCII
Is Remainder < 65
Push the remainder value in the stack
Pop the hs Code from the stack
Stop

Fig 5-1: Process flow diagram hs code generation for bit(0 -9)

For backtracking ASCII value of the each alphabet in h_s code is used. The first three bits from the MSB are separately read and summation of ASCII values of the remaining bits (9th bit to LSB) will give the weighted ASCII code which will be converted back to the decimal equivalent. The 10th bit is checked for the ASCII code. If the 10th bit ASCII code is 79 or 83, only then the value is added to summation of 9 bits (Figure 4-2). Frequency f(p) of each p € Palpha{} presented in the h_s code help in searching the exact code at improved rate. One of the most important realization is the frequency of G, if the f(G) where G € Palpha{}is 1 the h_s code from 44-53 is encountered. Algorithm for both h_s code backtracking the location is as follows:

Algorithm 5-2: Retrieving location from h_s code

```
1: code ←  input()
2: j ←len(code)
3: for k in range(j):
4:      if k < 1 :
5:           sum + ← dict(code[k])
6:      elsif k =1:
7:           num1 ←int(code[1])
8:      else:
9:           num ← int(code[0])
10:          num  ←  num1*10 + num
11:          sum ← sum+num
```

The following process flow converting the location value into h_s code is presented in Figure5-2

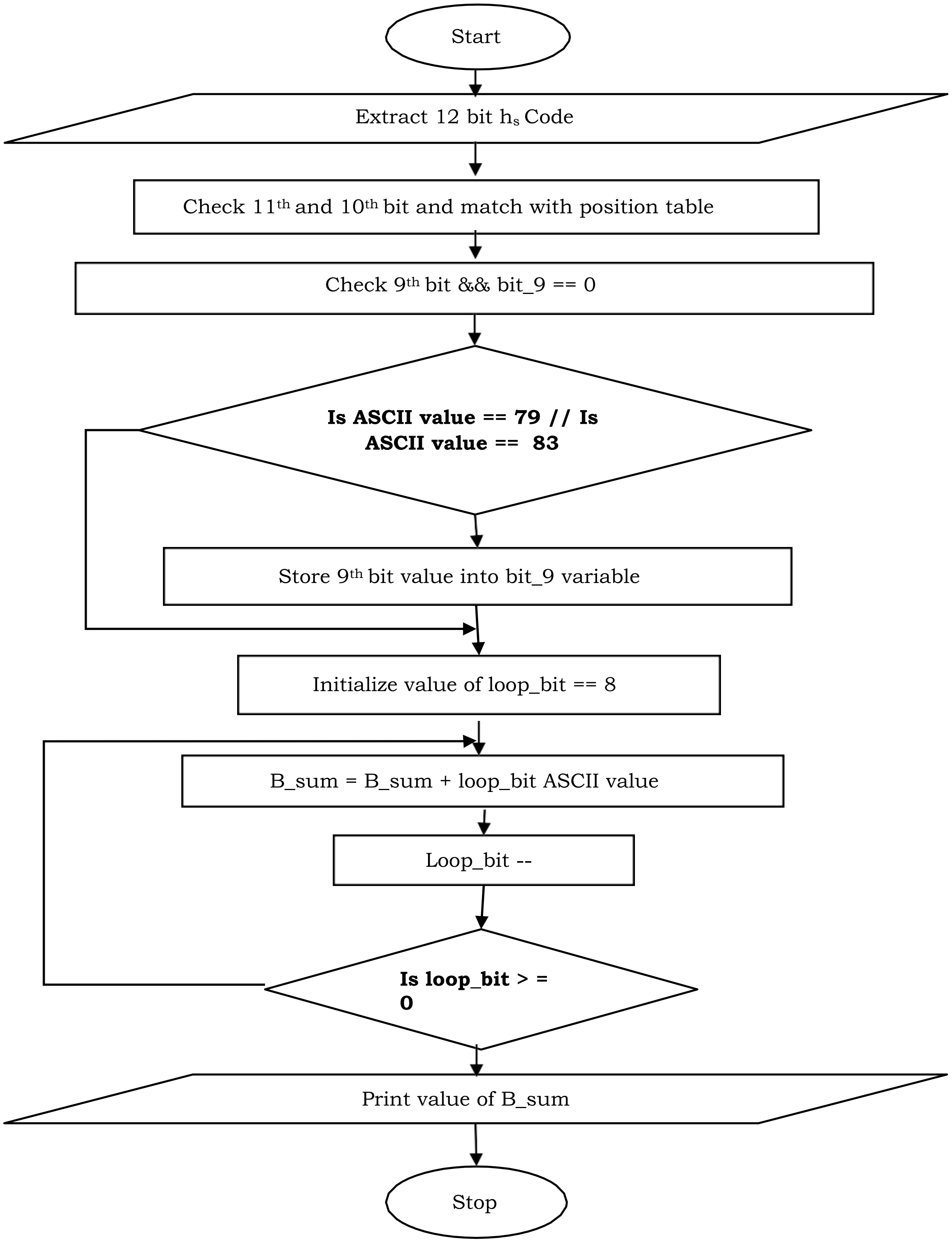

Start
Extract 12 bit h_s Code
Check 11th and 10th bit and match with position table
Check 9th bit && bit_9 == 0
Is ASCII value == 79 // Is ASCII value == 83
Store 9th bit value into bit_9 variable
Initialize value of loop_bit == 8
B_sum = B_sum + loop_bit ASCII value
Loop_bit --
Is loop_bit > = 0
Print value of B_sum
Stop

Fig 5-2: Process flow diagram for Back tracking from h_s code

These “h_s code” for all the number from 0 to 90 is represented in the table 5-2. The 10 bit code is further compressed and change to 8 bit code (table 5-3) and finally converted into a consolidated decimal code which is reconverted into an ASCII keyword. This whole process of converting the location into unique keyword is the heart of the proposed spatial search. The following section will demonstrate various processes undertaken to convert consolidate location code.

Table5-2: h_s code 0-90 number

Number	**W_ASCII**	**Hs code**	11	10	9	8	7	6	5	4	3	2	1	0
0	**528**	GCGCICC45			**Y**	G	C	G	C	I	C	C	4	5
1	529	GCICICC44			**Y**	G	C	I	C	I	C	C	4	4
2	530	GCICICC45			**Y**	G	C	I	C	I	C	C	4	5
3	531	GCICICC46			**Y**	G	C	I	C	I	C	C	4	6
4	532	GCICICC47			**Y**	G	C	I	C	I	C	C	4	7
5	533	ICICICC46			**Y**	I	C	I	C	I	C	C	4	6
6	534	ICICICC47			**Y**	I	C	I	C	I	C	C	4	7
7	535	ICICICC48			**Y**	I	C	I	C	I	C	C	4	8
8	536	ICICOCC43			**Y**	I	C	I	C	O	C	C	4	3
9	537	ICICOCC44			**Y**	I	C	I	C	O	C	C	4	4
10	538	ICICOCC45			**Y**	I	C	I	C	O	C	C	4	5
11	539	ICICOCC46			**Y**	I	C	I	C	O	C	C	4	6
12	540	ICICOCC47			**Y**	I	C	I	C	O	C	C	4	7
13	541	ICICOCC48			**Y**	I	C	I	C	O	C	C	4	8
14	542	ICOCOCC43			**Y**	I	C	O	C	O	C	C	4	3
15	543	ICOCOCC44			**Y**	I	C	O	C	O	C	C	4	4
16	544	ICOCOCC45			**Y**	I	C	O	C	O	C	C	4	5
17	545	ICOCOCC46			**Y**	I	C	O	C	O	C	C	4	6
18	546	ICOCOCC47			**Y**	I	C	O	C	O	C	C	4	7
19	547	ICOCOCC48			**Y**	I	C	O	C	O	C	C	4	8
20	548	OCOCOCC43			**Y**	O	C	O	C	O	C	C	4	3
21	549	OCOCOCC44			**Y**	O	C	O	C	O	C	C	4	4
22	550	OCOCOCC45			**Y**	O	C	O	C	O	C	C	4	5
23	551	OCOCOCC46			**Y**	O	C	O	C	O	C	C	4	6
24	552	OCOCOCC47			**Y**	O	C	O	C	O	C	C	4	7
25	553	OCOCOCC48			**Y**	O	C	O	C	O	C	C	4	8
26	554	OCOCOCC49			**Y**	O	C	O	C	O	C	C	4	9
27	555	OCOCOCC50			**Y**	O	C	O	C	O	C	C	5	0
28	556	OCOCOCC51			**Y**	O	C	O	C	O	C	C	5	1
29	557	OCOCOCC52			**Y**	O	C	O	C	O	C	C	5	2

30	558	OCOCOCC53			Y	O	C	O	C	O	C	C	5	3
31	559	OCOCOCC54			Y	O	C	O	C	O	C	C	5	4
32	560	OCOCOCC55			Y	O	C	O	C	O	C	C	5	5
33	561	OCOCOCC56			Y	O	C	O	C	O	C	C	5	6
34	562	OCOCOCC57			Y	O	C	O	C	O	C	C	5	7
35	563	OCOCOCC58			Y	O	C	O	C	O	C	C	5	8
36	564	OCOCOCC59			Y	O	C	O	C	O	C	C	5	9
37	565	OCOCOCC60			Y	O	C	O	C	O	C	C	6	0
38	566	OCOCOCC61			Y	O	C	O	C	O	C	C	6	1
39	567	OCOCOCC62			Y	O	C	O	C	O	C	C	6	2
40	568	OCOCOCC63			Y	O	C	O	C	O	C	C	6	3
41	569	OCOCSCC60			Y	O	C	O	C	S	C	C	6	0
42	570	OCOCSCC61			Y	O	C	O	C	S	C	C	6	1
43	571	OCOCSCC62			Y	O	C	O	C	S	C	C	6	2
44	572	OCOCSCC63			Y	O	C	O	C	S	C	C	6	3
45	573	OCOCSCC64			Y	O	C	O	C	S	C	C	6	4
46	574	OCOCSCC65			Y	O	C	O	C	S	C	C	6	5
47	575	OCOCSCC66			Y	O	C	O	C	S	C	C	6	6
48	576	OCOCSCCC0			I	O	C	O	C	S	C	C	C	0
49	577	OCOCSCCC1			I	O	C	O	C	S	C	C	C	1
50	578	OCOCSCCC2			I	O	C	O	C	S	C	C	C	2
51	579	OCOCSCCC3			I	O	C	O	C	S	C	C	C	3
52	580	OCOCSCCC4			I	O	C	O	C	S	C	C	C	4
53	581	OCSCSCCC1			I	O	C	S	C	S	C	C	C	1
54	582	OGOCSCCC2			I	O	G	O	C	S	C	C	C	2
55	583	OGOCSCCC3			I	O	G	O	C	S	C	C	C	3
56	584	OGOCSCCC4			I	O	G	O	C	S	C	C	C	4
57	585	OGSCSCCC1			I	O	G	S	C	S	C	C	C	1
58	586	OGCGYCCC8			I	O	G	C	G	Y	C	C	C	8
59	587	OGCGYCCC9			I	O	G	C	G	Y	C	C	C	9
60	588	OGCGYCCC10			O	G	C	G	Y	C	C	C	1	0
61	589	OGCGYCCC11			O	G	C	G	Y	C	C	C	1	1
62	590	OGCGCGCC30			O	G	C	G	C	G	C	C	3	0
63	591	SGCGYCCC9			I	S	G	C	G	Y	C	C	C	9
64	592	SGCGYCCC10			S	G	C	G	Y	C	C	C	1	0
65	593	SGCGYCCC11			S	G	C	G	Y	C	C	C	1	1
66	594	SGCGCGCC30			S	G	C	G	C	G	C	C	3	0
67	595	SGCGCGCC31			S	G	C	G	C	G	C	C	3	1
68	596	SGCGCGCC32			S	G	C	G	C	G	C	C	3	2
69	597	SGCGCGCC33			S	G	C	G	C	G	C	C	3	3
70	598	SGCGCGCC34			S	G	C	G	C	G	C	C	3	4

71	599	SGCGCGCC35			**S**	G	C	G	C	G	C	C	3	5
72	600	SGCGCGCC36			**S**	G	C	G	C	G	C	C	3	6
73	601	SGCGCGCC37			**S**	G	C	G	C	G	C	C	3	7
74	602	SGCGCGCC38			**S**	G	C	G	C	G	C	C	3	8
75	603	SGCGCGCC39			**S**	G	C	G	C	G	C	C	3	9
76	604	SGCGCGCC40			**S**	G	C	G	C	G	C	C	4	0
77	605	SGCGCGCC41			**S**	G	C	G	C	G	C	C	4	1
78	606	SGCGCGCC42			**S**	G	C	G	C	G	C	C	4	2
79	607	SGCGCICC41			**S**	G	C	G	C	I	C	C	4	1
80	608	SGCGCICC42			**S**	G	C	G	C	I	C	C	4	2
81	609	SGCGCICC43			**S**	G	C	G	C	I	C	C	4	3
82	610	SGCGCICC44			**S**	G	C	G	C	I	C	C	4	4
83	611	SGCGCICC45			**S**	G	C	G	C	I	C	C	4	5
84	612	SGCICICC44			**S**	G	C	I	C	I	C	C	4	4
85	613	SGCICICC45			**S**	G	C	I	C	I	C	C	4	5
86	614	SGCICICC46			**S**	G	C	I	C	I	C	C	4	6
87	615	SGCICICC47			**S**	G	C	I	C	I	C	C	4	7
88	616	SICICICC46			**S**	I	C	I	C	I	C	C	4	6
89	617	SICICICC47			**S**	I	C	I	C	I	C	C	4	7
90	618	SICICICC48			**S**	I	C	I	C	I	C	C	4	8

These h_s code for all the number from 0 to 90 is represented in the table5-3. The 10 bit code is further compressed and change to 8 bit code and finally converted into a consolidated decimal code which is reconverted into an ASCII keyword

Table5-3: h_s code 0-90 (8 bit)

Number	**W_ASCII**	**Hs code**	7	6	5	4	3	2	1	0
0	**528**	GCGCICC45	G	C	G	C	I	Y	4	5
1	529	GCICICC44	G	C	I	C	I	Y	4	4
2	530	GCICICC45	G	C	I	C	I	Y	4	5
3	531	GCICICC46	G	C	I	C	I	Y	4	6
4	532	GCICICC47	G	C	I	C	I	Y	4	7
5	533	ICICICC46	I	C	I	C	I	Y	4	6
6	534	ICICICC47	I	C	I	C	I	Y	4	7
7	535	ICICICC48	I	C	I	C	I	Y	4	8
8	536	ICICOCC43	I	C	I	C	O	Y	4	3
9	537	ICICOCC44	I	C	I	C	O	Y	4	4
10	538	ICICOCC45	I	C	I	C	O	Y	4	5
11	539	ICICOCC46	I	C	I	C	O	Y	4	6

12	540	ICICOCC47	I	C	I	C	O	Y	4	7
13	541	ICICOCC48	I	C	I	C	O	Y	4	8
14	542	ICOCOCC43	I	C	O	C	O	Y	4	3
15	543	ICOCOCC44	I	C	O	C	O	Y	4	4
16	544	ICOCOCC45	I	C	O	C	O	Y	4	5
17	545	ICOCOCC46	I	C	O	C	O	Y	4	6
18	546	ICOCOCC47	I	C	O	C	O	Y	4	7
19	547	ICOCOCC48	I	C	O	C	O	Y	4	8
20	548	OCOCOCC43	O	C	O	C	O	Y	4	3
21	549	OCOCOCC44	O	C	O	C	O	Y	4	4
22	550	OCOCOCC45	O	C	O	C	O	Y	4	5
23	551	OCOCOCC46	O	C	O	C	O	Y	4	6
24	552	OCOCOCC47	O	C	O	C	O	Y	4	7
25	553	OCOCOCC48	O	C	O	C	O	Y	4	8
26	554	OCOCOCC49	O	C	O	C	O	Y	4	9
27	555	OCOCOCC50	O	C	O	C	O	Y	5	0
28	556	OCOCOCC51	O	C	O	C	O	Y	5	1
29	557	OCOCOCC52	O	C	O	C	O	Y	5	2
30	558	OCOCOCC53	O	C	O	C	O	Y	5	3
31	559	OCOCOCC54	O	C	O	C	O	Y	5	4
32	560	OCOCOCC55	O	C	O	C	O	Y	5	5
33	561	OCOCOCC56	O	C	O	C	O	Y	5	6
34	562	OCOCOCC57	O	C	O	C	O	Y	5	7
35	563	OCOCOCC58	O	C	O	C	O	Y	5	8
36	564	OCOCOCC59	O	C	O	C	O	Y	5	9
37	565	OCOCOCC60	O	C	O	C	O	Y	6	0
38	566	OCOCOCC61	O	C	O	C	O	Y	6	1
39	567	OCOCOCC62	O	C	O	C	O	Y	6	2
40	568	OCOCOCC63	O	C	O	C	O	Y	6	3
41	569	OCOCSCC60	O	C	O	C	S	Y	6	0
42	570	OCOCSCC61	O	C	O	C	S	Y	6	1
43	571	OCOCSCC62	O	C	O	C	S	Y	6	2
44	572	OCOCSCC63	O	C	O	C	S	Y	6	3
45	573	OCOCSCC64	O	C	O	C	S	Y	6	4
46	574	OCOCSCC65	O	C	O	C	S	Y	6	5
47	575	OCOCSCC66	O	C	O	C	S	Y	6	6
48	576	OCOCSCCC0	O	C	O	C	S	I	C	0
49	577	OCOCSCCC1	O	C	O	C	S	I	C	1
50	578	OCOCSCCC2	O	C	O	C	S	I	C	2
51	579	OCOCSCCC3	O	C	O	C	S	I	C	3
52	580	OCOCSCCC4	O	C	O	C	S	I	C	4

53	581	OCSCSCCC1	O	C	S	C	S	I	C	1
54	582	OGOCSCCC2	O	G	O	C	S	I	C	2
55	583	OGOCSCCC3	O	G	O	C	S	I	C	3
56	584	OGOCSCCC4	O	G	O	C	S	I	C	4
57	585	OGSCSCCC1	O	G	S	C	S	I	C	1
58	586	OGCGYCCC8	O	G	C	G	Y	I	C	8
59	587	OGCGYCCC9	O	G	C	G	Y	I	C	9
60	588	OGCGYCCC10	G	C	G	Y	C	O	1	0
61	589	OGCGYCCC11	G	C	G	Y	C	O	1	1
62	590	OGCGCGCC30	G	C	G	C	G	O	3	0
63	591	SGCGYCCC9	S	G	C	G	Y	I	C	9
64	592	SGCGYCCC10	G	C	G	Y	C	S	1	0
65	593	SGCGYCCC11	G	C	G	Y	C	S	1	1
66	594	SGCGCGCC30	G	C	G	C	G	S	3	0
67	595	SGCGCGCC31	G	C	G	C	G	S	3	1
68	596	SGCGCGCC32	G	C	G	C	G	S	3	2
69	597	SGCGCGCC33	G	C	G	C	G	S	3	3
70	598	SGCGCGCC34	G	C	G	C	G	S	3	4
71	599	SGCGCGCC35	G	C	G	C	G	S	3	5
72	600	SGCGCGCC36	G	C	G	C	G	S	3	6
73	601	SGCGCGCC37	G	C	G	C	G	S	3	7
74	602	SGCGCGCC38	G	C	G	C	G	S	3	8
75	603	SGCGCGCC39	G	C	G	C	G	S	3	9
76	604	SGCGCGCC40	G	C	G	C	G	S	4	0
77	605	SGCGCGCC41	G	C	G	C	G	S	4	1
78	606	SGCGCGCC42	G	C	G	C	G	S	4	2
79	607	SGCGCICC41	G	C	G	C	I	S	4	1
80	608	SGCGCICC42	G	C	G	C	I	S	4	2
81	609	SGCGCICC43	G	C	G	C	I	S	4	3
82	610	SGCGCICC44	G	C	G	C	I	S	4	4
83	611	SGCGCICC45	G	C	G	C	I	S	4	5
84	612	SGCICICC44	G	C	I	C	I	S	4	4
85	613	SGCICICC45	G	C	I	C	I	S	4	5
86	614	SGCICICC46	G	C	I	C	I	S	4	6
87	615	SGCICICC47	G	C	I	C	I	S	4	7
88	616	SICICICC46	I	C	I	C	I	S	4	6
89	617	SICICICC47	I	C	I	C	I	S	4	7
90	618	SICICICC48	I	C	I	C	I	S	4	8

From these 8 bit “h_s code” further similarity and repetition of the code pair are identified. Hence a pair of bit 7 – 2 and quad pair of 6-5-4-3 are identified and represented with hexadecimal code. The identified codes and their respective hexadecimal code is represented in table 5-4

Table5-4: h_s code pairs and their hexadecimal code

S.No	Pair	Hexa Code	Quad pair	Hexa Code
1	GY	A9	CGCI	A1
2	GO	A8	CGCY	A2
3	GS	A7	CGCG	A3
4	IY	B9	CICI	B1
5	IS	B8	CICO	B2
6	OY	C9	COCO	C1
7	OI	C8	COCS	C2
8	SI	D9	CSCS	CC
9			GOCS	D1
10			GSCS	E1
11			GCGY	F1

Now the revised h_s code after replacing the pairs with hexa-pairs is shown in table 6. Another conversion is taken care in the form number 48-59 and 63 where the pair in bit 1 and 0 is converted into number by subtracting decimal value of bit o from 67 and the remainder two bit decimal number is stored in bit 1 and 0. The algorithm for encoding 12 bits h_s code in 6 bits hexadecimal code is as follows:

Algorithm5-3: Encoding 12 bit h_s code into 6 bit code

```
1: h← input(“enter hs code”,)
2: t ←    h[-9:-8]
3: m ←    h[-8:-4]
4: d ←    h[-1:-0]
5: nh ←    m+t+ char(d)
6: p1    ←    nh[7:6]
```

```
7: p2    ←   nh[5:4]
8: p3    ←   nh[3:2]
9: pd    ←   nh[1:0]
10: x    ←   nh[7]
11: y    ←   nh[2]
12: hd1  ←   x+y
13: hx1  ←   dist1.get("hd1")
14: hd2  ←   nh[6:3]
15: hx2  ←   dist2.get("hd2")
16: hxhs      ←   hx1+hx2+pd
```

Table5-5: h_s code and their respective decimal equivalent

Number	W_ASCII	7	6	5	4	3	2	1	0	h s code	Decimal
0	528	G	C	G	C	I	Y	4	5	A9A145	11116869
1	529	G	C	I	C	I	Y	4	4	A9B144	11120964
2	530	G	C	I	C	I	Y	4	5	A9B145	11120965
3	531	G	C	I	C	I	Y	4	6	A9B146	11120966
4	532	G	C	I	C	I	Y	4	7	A9B147	11120967
5	533	I	C	I	C	I	Y	4	6	B9B146	12169542
6	534	I	C	I	C	I	Y	4	7	B9B147	12169543
7	535	I	C	I	C	I	Y	4	8	B9B148	12169544
8	536	I	C	I	C	O	Y	4	3	B9B243	12169795
9	537	I	C	I	C	O	Y	4	4	B9B244	12169796
10	538	I	C	I	C	O	Y	4	5	B9B245	12169797
11	539	I	C	I	C	O	Y	4	6	B9B246	12169798
12	540	I	C	I	C	O	Y	4	7	B9B247	12169799
13	541	I	C	I	C	O	Y	4	8	B9B248	12169800
14	542	I	C	O	C	O	Y	4	3	B9C143	12173635
15	543	I	C	O	C	O	Y	4	4	B9C144	12173636

16	544	I	C	O	C	O	Y	4	5	B9C145	12173637
17	545	I	C	O	C	O	Y	4	6	B9C146	12173638
18	546	I	C	O	C	O	Y	4	7	B9C147	12173639
19	547	I	C	O	C	O	Y	4	8	B9C148	12173640
20	548	O	C	O	C	O	Y	4	3	C9C143	13222211
21	549	O	C	O	C	O	Y	4	4	C9C144	13222212
22	550	O	C	O	C	O	Y	4	5	C9C145	13222213
23	551	O	C	O	C	O	Y	4	6	C9C146	13222214
24	552	O	C	O	C	O	Y	4	7	C9C147	13222215
25	553	O	C	O	C	O	Y	4	8	C9C148	13222216
26	554	O	C	O	C	O	Y	4	9	C9C149	13222217
27	555	O	C	O	C	O	Y	5	0	C9C150	13222224
28	556	O	C	O	C	O	Y	5	1	C9C151	13222225
29	557	O	C	O	C	O	Y	5	2	C9C152	13222226
30	558	O	C	O	C	O	Y	5	3	C9C153	13222227
31	559	O	C	O	C	O	Y	5	4	C9C154	13222228
32	560	O	C	O	C	O	Y	5	5	C9C155	13222229
33	561	O	C	O	C	O	Y	5	6	C9C156	13222230
34	562	O	C	O	C	O	Y	5	7	C9C157	13222231
35	563	O	C	O	C	O	Y	5	8	C9C158	13222232
36	564	O	C	O	C	O	Y	5	9	C9C159	13222233
37	565	O	C	O	C	O	Y	6	0	C9C160	13222240
38	566	O	C	O	C	O	Y	6	1	C9C161	13222241
39	567	O	C	O	C	O	Y	6	2	C9C162	13222242
40	568	O	C	O	C	O	Y	6	3	C9C163	13222243
41	569	O	C	O	C	S	Y	6	0	C9C260	13222496
42	570	O	C	O	C	S	Y	6	1	C9C261	13222497
43	571	O	C	O	C	S	Y	6	2	C9C262	13222498
44	572	O	C	O	C	S	Y	6	3	C9C263	13222499
45	573	O	C	O	C	S	Y	6	4	C9C264	13222500
46	574	O	C	O	C	S	Y	6	5	C9C265	13222501
47	575	O	C	O	C	S	Y	6	6	C9C266	13222502
48	576	O	C	O	C	S	I	6	7	C8C267	13156967
49	577	O	C	O	C	S	I	6	6	C8C266	13156966
50	578	O	C	O	C	S	I	6	5	C8C265	13156965
51	579	O	C	O	C	S	I	6	4	C8C264	13156964
52	580	O	C	O	C	S	I	6	3	C8C263	13156963
53	581	O	C	S	C	S	I	6	6	C8CC66	13159526
54	582	O	G	O	C	S	I	6	5	C8D165	13160805

55	583	O	G	O	C	S	I	6	4	C8D164	13160804
56	584	O	G	O	C	S	I	6	3	C8D163	13160803
57	585	O	G	S	C	S	I	6	6	C8E166	13164902
58	586	O	G	C	G	Y	I	5	9	C8F159	13168985
59	587	O	G	C	G	Y	I	5	8	C8F158	13168984
60	588	G	C	G	Y	C	O	1	0	A8A210	11051536
61	589	G	C	G	Y	C	O	1	1	A8A211	11051537
62	590	G	C	G	C	G	O	3	0	A8A330	11051824
63	591	S	G	C	G	Y	I	5	8	D9F158	14283096
64	592	G	C	G	Y	C	S	1	0	A7A210	10986000
65	593	G	C	G	Y	C	S	1	1	A7A211	10986001
66	594	G	C	G	C	G	S	3	0	A7A330	10986288
67	595	G	C	G	C	G	S	3	1	A7A331	10986289
68	596	G	C	G	C	G	S	3	2	A7A332	10986290
69	597	G	C	G	C	G	S	3	3	A7A333	10986291
70	598	G	C	G	C	G	S	3	4	A7A334	10986292
71	599	G	C	G	C	G	S	3	5	A7A335	10986293
72	600	G	C	G	C	G	S	3	6	A7A336	10986294
73	601	G	C	G	C	G	S	3	7	A7A337	10986295
74	602	G	C	G	C	G	S	3	8	A7A338	10986296
75	603	G	C	G	C	G	S	3	9	A7A339	10986297
76	604	G	C	G	C	G	S	4	0	A7A340	10986304
77	605	G	C	G	C	G	S	4	1	A7A341	10986305
78	606	G	C	G	C	G	S	4	2	A7A342	10986306
79	607	G	C	G	C	I	S	4	1	A7A141	10985793
80	608	G	C	G	C	I	S	4	2	A7A142	10985794
81	609	G	C	G	C	I	S	4	3	A7A143	10985795
82	610	G	C	G	C	I	S	4	4	A7A144	10985796
83	611	G	C	G	C	I	S	4	5	A7A145	10985797
84	612	G	C	I	C	I	S	4	4	A7B144	10989892
85	613	G	C	I	C	I	S	4	5	A7B145	10989893
86	614	G	C	I	C	I	S	4	6	A7B146	10989894
87	615	G	C	I	C	I	S	4	7	A7B147	10989895
88	616	I	C	I	C	I	S	4	6	B8B146	12104006
89	617	I	C	I	C	I	S	4	7	B8B147	12104007
90	618	I	C	I	C	I	S	4	8	B8B148	12104008

Finally after convert a number into hs code and finally to a customized decimal code, the arithmetic operation will be performed on all the four partition of sexagesimal value i.e. dd°.mm’ss.ss’’. The final compressed hexadecimal code of given latitude location 1°.1’1.1” to 1°.1’1.59” is shown in table 5-6.

Table5-6: Final h_s code of location

S.No	DD			MM			SS			.SS			Total	Hexadecimal Code
1	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	44483856	**2A6C510**
2	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**2**	A9B145	11120965	44483857	**2A6C511**
3	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**3**	A9B146	11120966	44483858	**2A6C512**
4	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**4**	A9B147	11120967	44483859	**2A6C513**
5	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**5**	B9B146	12169542	45532434	**2B6C512**
6	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**6**	B9B147	12169543	45532435	**2B6C513**
7	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**7**	B9B148	12169544	45532436	**2B6C514**
8	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**8**	B9B243	12169795	45532687	**2B6C60F**
9	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**9**	B9B244	12169796	45532688	**2B6C610**
10	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**10**	B9B245	12169797	45532689	**2B6C611**
11	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**11**	B9B246	12169798	45532690	**2B6C612**
12	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**12**	B9B247	12169799	45532691	**2B6C613**
13	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**13**	B9B248	12169800	45532692	**2B6C614**
14	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**14**	B9C143	12173635	45536527	**2B6D50F**
15	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**15**	B9C144	12173636	45536528	**2B6D510**
16	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**16**	B9C145	12173637	45536529	**2B6D511**
17	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**17**	B9C146	12173638	45536530	**2B6D512**
18	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**18**	B9C147	12173639	45536531	**2B6D513**
19	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**19**	B9C148	12173640	45536532	**2B6D514**
20	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**20**	C9C143	13222211	46585103	**2C6D50F**
21	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**21**	C9C144	13222212	46585104	**2C6D510**

22	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**2 2**	C9C 145	1322 2213	4658 5105	**2C6D511**
23	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**2 3**	C9C 146	1322 2214	4658 5106	**2C6D512**
24	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**2 4**	C9C 147	1322 2215	4658 5107	**2C6D513**
25	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**2 5**	C9C 148	1322 2216	4658 5108	**2C6D514**
26	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**2 6**	C9C 149	1322 2217	4658 5109	**2C6D515**
27	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**2 7**	C9C 150	1322 2224	4658 5116	**2C6D51C**
28	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**2 8**	C9C 151	1322 2225	4658 5117	**2C6D51D**
29	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**2 9**	C9C 152	1322 2226	4658 5118	**2C6D51E**
30	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**3 0**	C9C 153	1322 2227	4658 5119	**2C6D51F**
31	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**3 1**	C9C 154	1322 2228	4658 5120	**2C6D520**
32	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**3 2**	C9C 155	1322 2229	4658 5121	**2C6D521**
33	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**3 3**	C9C 156	1322 2230	4658 5122	**2C6D522**
34	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**3 4**	C9C 157	1322 2231	4658 5123	**2C6D523**
35	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**3 5**	C9C 158	1322 2232	4658 5124	**2C6D524**
36	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**3 6**	C9C 159	1322 2233	4658 5125	**2C6D525**
37	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**3 7**	C9C 160	1322 2240	4658 5132	**2C6D52C**
38	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**3 8**	C9C 161	1322 2241	4658 5133	**2C6D52D**
39	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**3 9**	C9C 162	1322 2242	4658 5134	**2C6D52E**
40	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**4 0**	C9C 163	1322 2243	4658 5135	**2C6D52F**
41	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**4 1**	C9C 260	1322 2496	4658 5388	**2C6D62C**
42	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**4 2**	C9C 261	1322 2497	4658 5389	**2C6D62D**
43	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**4 3**	C9C 262	1322 2498	4658 5390	**2C6D62E**
44	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**4 4**	C9C 263	1322 2499	4658 5391	**2C6D62F**
45	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**4 5**	C9C 264	1322 2500	4658 5392	**2C6D630**
46	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**4 6**	C9C 265	1322 2501	4658 5393	**2C6D631**
47	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**4 7**	C9C 266	1322 2502	4658 5394	**2C6D632**
48	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**1**	A9B 144	1112 0964	**4 8**	C8C 267	1315 6967	4651 9859	**2C5D633**

49	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**49**	C8C266	13156966	46519858	**2C5D632**
50	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**50**	C8C265	13156965	46519857	**2C5D631**
51	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**51**	C8C264	13156964	46519856	**2C5D630**
52	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**52**	C8C263	13156963	46519855	**2C5D62F**
53	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**53**	C8CC66	13159526	46522418	**2C5E032**
54	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**54**	C8D165	13160805	46523697	**2C5E531**
55	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**55**	C8D164	13160804	46523696	**2C5E530**
56	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**56**	C8D163	13160803	46523695	**2C5E52F**
57	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**57**	C8E166	13164902	46527794	**2C5F532**
58	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**58**	C8F159	13168985	46531877	**2C60525**
59	**1**	A9B144	11120964	**1**	A9B144	11120964	**1**	A9B144	11120964	**59**	C8F158	13168984	46531876	**2C60524**

The final compressed hexadecimal code of given latitude location 20°.45'30.1" to 20°.45'30.59" latitude and longitude 10°.40'30.20" is shown in table 5-7.

Table5-7: Consolidate 7 bit hs code

S.No	**Latitude**	**Decimal Value**	**Hx Code(Lat)**	**Longitude**	**Decimal Value**	**Hx Code(Long)**	**Sumation**	**Hx Code**
1	20^.45'30.1"	50787902	306F63E	10^.40'20.59"	51783235	3162643	102571137	61D1C81
2	20^.45'30.2"	50787903	306F63F	10^.40'20.59"	51783235	3162643	102571138	61D1C82
3	20^.45'30.3"	50787904	306F640	10^.40'20.59"	51783235	3162643	102571139	61D1C83
4	20^.45'30.4"	50787905	306F641	10^.40'20.59"	51783235	3162643	102571140	61D1C84
5	20^.45'30.5"	51836480	316F640	10^.40'20.59"	51783235	3162643	103619715	62D1C83
6	20^.45'30.6"	51836481	316F641	10^.40'20.59"	51783235	3162643	103619716	62D1C84
7	20^.45'30.7"	51836482	316F642	10^.40'20.59"	51783235	3162643	103619717	62D1C85
8	20^.45'30.8"	51836733	316F73D	10^.40'20.59"	51783235	3162643	103619968	62D1D80
9	20^.45'30.9"	51836734	316F73E	10^.40'20.59"	51783235	3162643	103619969	62D1D81
10	20^.45'30.10"	51836735	316F73F	10^.40'20.59"	51783235	3162643	103619970	62D1D82

11	20^.45'30.11"	51836736	316F740	10^.40'20.59"	51783235	3162643	103619971	62D1D83
12	20^.45'30.12"	51836737	316F741	10^.40'20.59"	51783235	3162643	103619972	62D1D84
13	20^.45'30.13"	51836738	316F742	10^.40'20.59"	51783235	3162643	103619973	62D1D85
14	20^.45'30.14"	51840573	317063D	10^.40'20.59"	51783235	3162643	103623808	62D2C80
15	20^.45'30.15"	51840574	317063E	10^.40'20.59"	51783235	3162643	103623809	62D2C81
16	20^.45'30.16"	51840575	317063F	10^.40'20.59"	51783235	3162643	103623810	62D2C82
17	20^.45'30.17"	51840576	3170640	10^.40'20.59"	51783235	3162643	103623811	62D2C83
18	20^.45'30.18"	51840577	3170641	10^.40'20.59"	51783235	3162643	103623812	62D2C84
19	20^.45'30.19"	51840578	3170642	10^.40'20.59"	51783235	3162643	103623813	62D2C85
20	20^.45'30.20"	52889149	327063D	10^.40'20.59"	51783235	3162643	104672384	63D2C80
21	20^.45'30.21"	52889150	327063E	10^.40'20.59"	51783235	3162643	104672385	63D2C81
22	20^.45'30.22"	52889151	327063F	10^.40'20.59"	51783235	3162643	104672386	63D2C82
23	20^.45'30.23"	52889152	3270640	10^.40'20.59"	51783235	3162643	104672387	63D2C83
24	20^.45'30.24"	52889153	3270641	10^.40'20.59"	51783235	3162643	104672388	63D2C84
25	20^.45'30.25"	52889154	3270642	10^.40'20.59"	51783235	3162643	104672389	63D2C85
26	20^.45'30.26"	52889155	3270643	10^.40'20.59"	51783235	3162643	104672390	63D2C86
27	20^.45'30.27"	52889162	327064A	10^.40'20.59"	51783235	3162643	104672397	63D2C8D
28	20^.45'30.28"	52889163	327064B	10^.40'20.59"	51783235	3162643	104672398	63D2C8E
29	20^.45'30.29"	52889164	327064C	10^.40'20.59"	51783235	3162643	104672399	63D2C8F
30	20^.45'30.30"	52889165	327064D	10^.40'20.59"	51783235	3162643	104672400	63D2C90
31	20^.45'30.31"	52889166	327064E	10^.40'20.59"	51783235	3162643	104672401	63D2C91
32	20^.45'30.32"	52889167	327064F	10^.40'20.59"	51783235	3162643	104672402	63D2C92
33	20^.45'30.33"	52889168	3270650	10^.40'20.59"	51783235	3162643	104672403	63D2C93
34	20^.45'30.34"	52889169	3270651	10^.40'20.59"	51783235	3162643	104672404	63D2C94
35	20^.45'30.35"	52889170	3270652	10^.40'20.59"	51783235	3162643	104672405	63D2C95
36	20^.45'30.36"	52889171	3270653	10^.40'20.59"	51783235	3162643	104672406	63D2C96
37	20^.45'30.37"	52889178	327065A	10^.40'20.59"	51783235	3162643	104672413	63D2C9D

38	20^.45'30.38"	52889179	327065B	10^.40'20.59"	51783235	3162643	104672414	63D2C9E
39	20^.45'30.39"	52889180	327065C	10^.40'20.59"	51783235	3162643	104672415	63D2C9F
40	20^.45'30.40"	52889181	327065D	10^.40'20.59"	51783235	3162643	104672416	63D2CA0
41	20^.45'30.41"	52889434	327075A	10^.40'20.59"	51783235	3162643	104672669	63D2D9D
42	20^.45'30.42"	52889435	327075B	10^.40'20.59"	51783235	3162643	104672670	63D2D9E
43	20^.45'30.43"	52889436	327075C	10^.40'20.59"	51783235	3162643	104672671	63D2D9F
44	20^.45'30.44"	52889437	327075D	10^.40'20.59"	51783235	3162643	104672672	63D2DA0
45	20^.45'30.45"	52889438	327075E	10^.40'20.59"	51783235	3162643	104672673	63D2DA1
46	20^.45'30.46"	52889439	327075F	10^.40'20.59"	51783235	3162643	104672674	63D2DA2
47	20^.45'30.47"	52889440	3270760	10^.40'20.59"	51783235	3162643	104672675	63D2DA3
48	20^.45'30.48"	52823905	3260761	10^.40'20.59"	51783235	3162643	104607140	63C2DA4
49	20^.45'30.49"	52823904	3260760	10^.40'20.59"	51783235	3162643	104607139	63C2DA3
50	20^.45'30.50"	52823903	326075F	10^.40'20.59"	51783235	3162643	104607138	63C2DA2
51	20^.45'30.51"	52823902	326075E	10^.40'20.59"	51783235	3162643	104607137	63C2DA1
52	20^.45'30.52"	52823901	326075D	10^.40'20.59"	51783235	3162643	104607136	63C2DA0
53	20^.45'30.53"	52826464	3261160	10^.40'20.59"	51783235	3162643	104609699	63C37A3
54	20^.45'30.54"	52827743	326165F	10^.40'20.59"	51783235	3162643	104610978	63C3CA2
55	20^.45'30.55"	52827742	326165E	10^.40'20.59"	51783235	3162643	104610977	63C3CA1
56	20^.45'30.56"	52827741	326165D	10^.40'20.59"	51783235	3162643	104610976	63C3CA0
57	20^.45'30.57"	52831840	3262660	10^.40'20.59"	51783235	3162643	104615075	63C4CA3
58	20^.45'30.58"	52835923	3263653	10^.40'20.59"	51783235	3162643	104619158	63C5C96
59	20^.45'30.59"	52835922	3263652	10^.40'20.59"	51783235	3162643	104619157	63C5C95

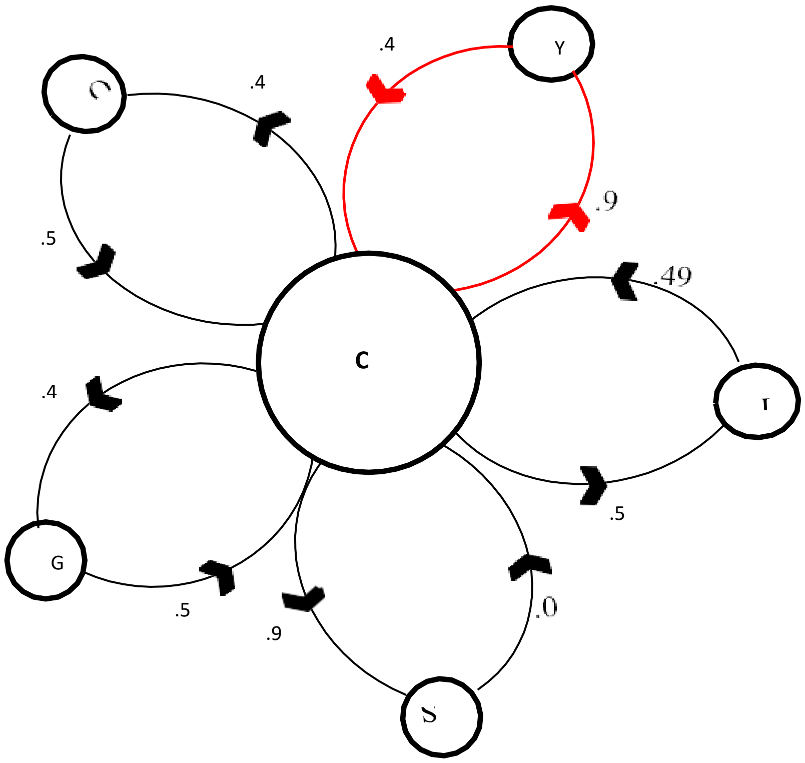
The process of h_s code encoding present us with a unique hexadecimal code which will be can summarize the location of any spatial object in study. These hexadecimal code will be stored in the column table where table key will help us in recalling back the original sexagesimal location of the object. These code plays a very crucial as well as important role in improving the manipulation rate of the spatial data. Traversing through these node one can easy identify the most suitable location that can be accessed under the given circumstance. After applying the indexer on to the location h_s code one will be able to produce the location indexed quad tree where only those node which are registered by the system under the predefined conditions will be available in the data structure and whose reference will be present in the pre-fetch. Hence on every query fired the system will check for the pre-fetch quad tree and will identified the required location meet the query keywords. And because multiple different h_s keyword will be present under as given cluster hence there will be no need to traverse through the whole location but only the location mentioned in that indexed keyword and their associated location.

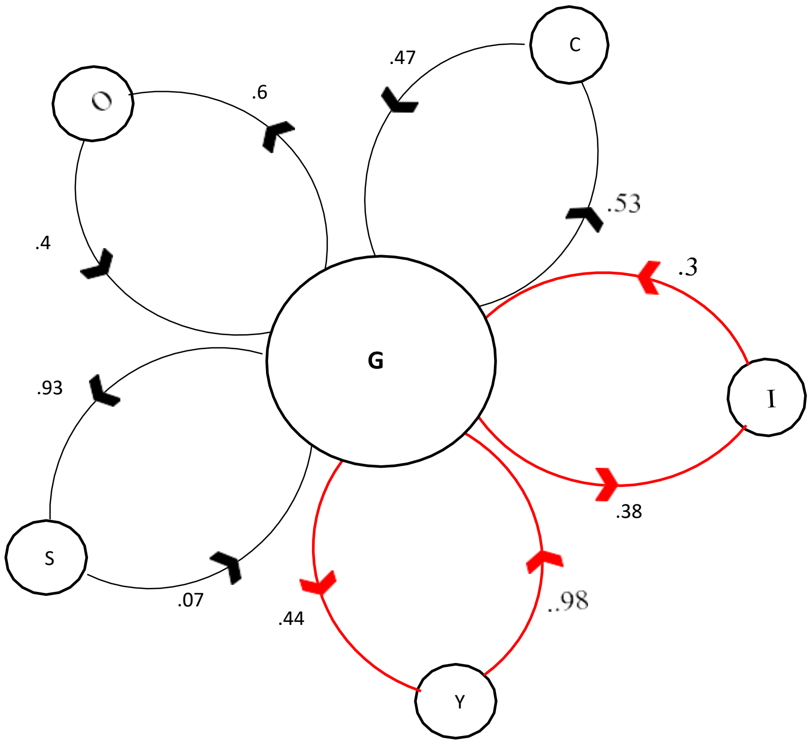
5.2 HIDDEN MARKOV MODEL FOR THE HS CODE

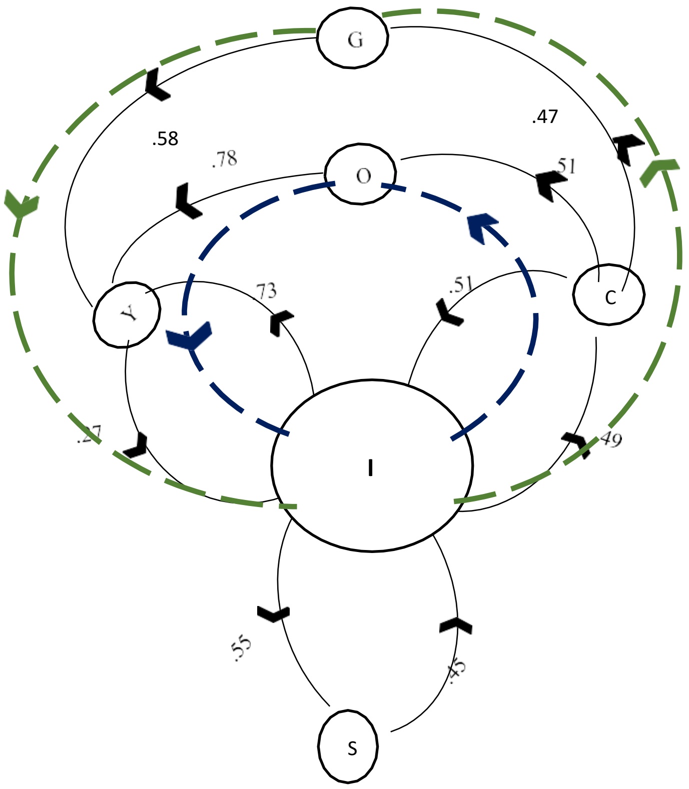
The generated h_s codes are group of four 12 bits number for each latitude and longitude location. These codes can be further simplified to produce single 7 bit cumulative code of the given location. During the process of encoding the h_s code into 7 bit hexadecimal code various different observation and hidden states are produces which thrives the probabilistic model of the generated location code.

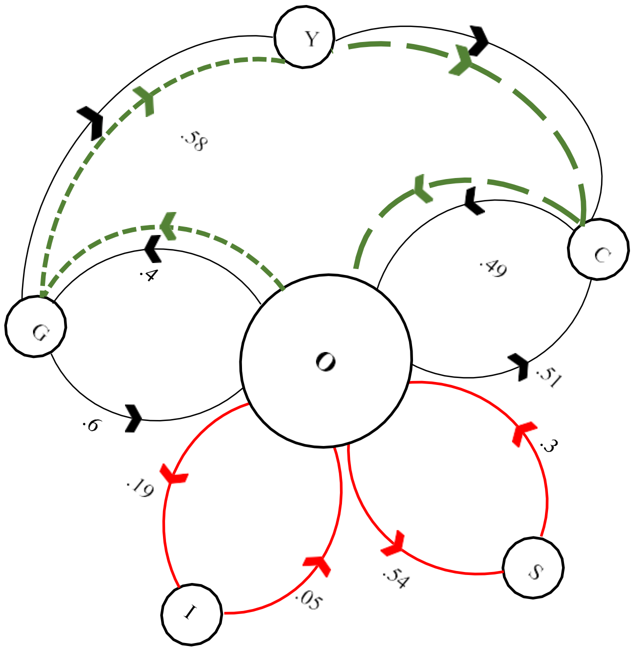
The converted hexadecimal code for each segment of the sexigeminal code will be converted followed by hexadecimal athematic to produce cumulative location hexadecimal code. List of p1, p2 and p3 will produce various state of the h_s code. Combination of these state and their transaction is one such evaluation problem that can be solved using Hidden Markova model (hmm).

A though deduction of the above created h_s codes, outline the following outcomes such as: firstly the number of observations are 90; secondly there are 17 state pair whose combination will produce an observation and thirdly the produced hidden markov model will work with 19 hidden state. The sequence of the state will produce the unique h_s code and its maximum probability to have a particular observation will be calculated by hmm. Alike any markov model, markov assumption is also applicable in this presented model i.e. for the prediction of future state only the present state matter, there is no correlation of past state with future state. Figure 5-4 present state transition probability of 6 original states forming the h_s code.


C
Y
.4
.9
.4
.5
.49
.5
.4
.5
.9
.0
G
S


C
.47
.53
O
.6
.4
G
.3
I
.38
.93
S
.07
.44
..98
Y


G
O
Y
C
I
S
.58
.78
.47
.51
.73
.51
.27
.49
.55
.45


Y
G
C
O
I
S
.58
.4
.49
.6
.51
.19
.3
.05
.54

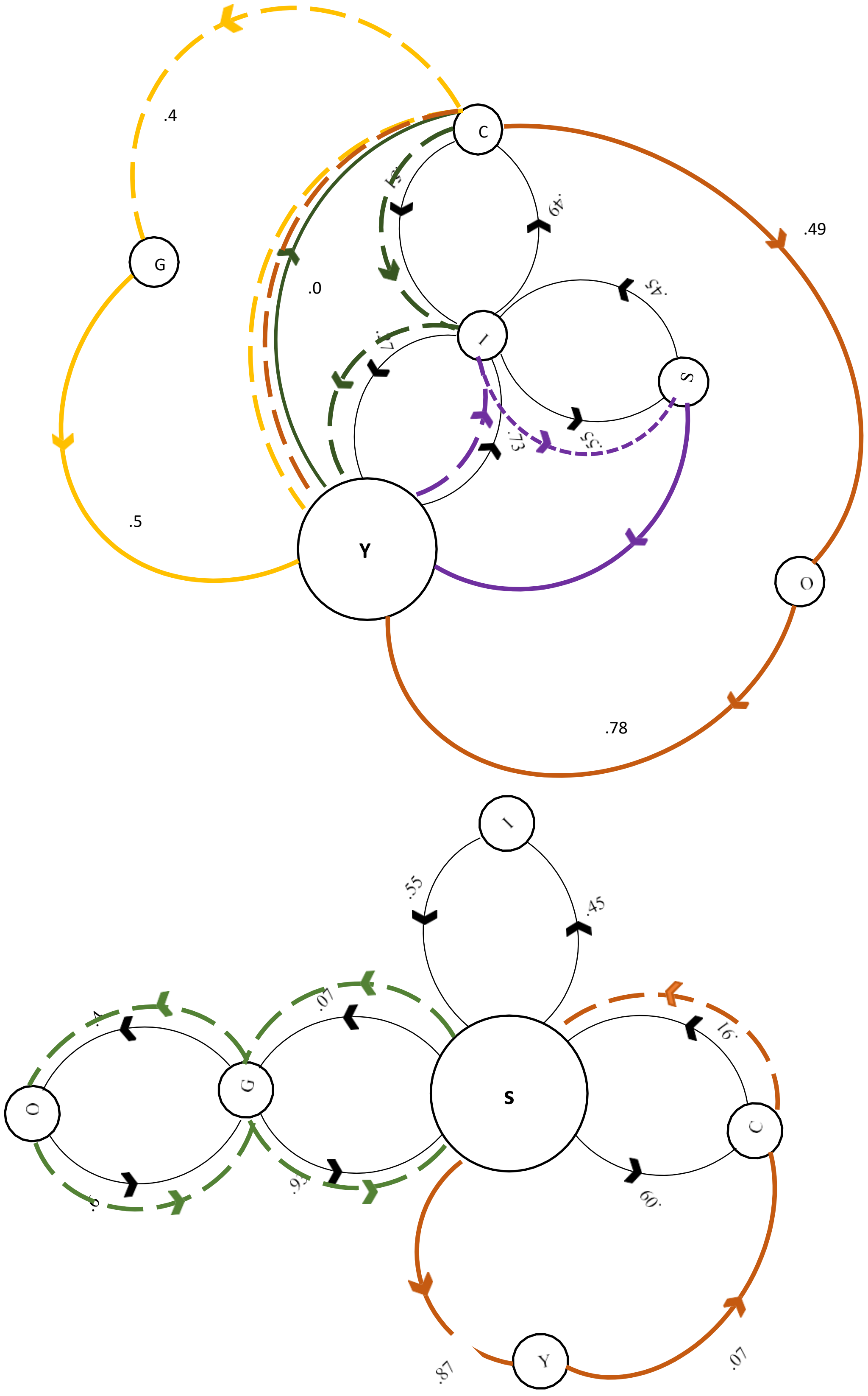


Fig5-4: State transition chat of different state in 8 bits h_s code

Pair of these states will formulate the 17 unique state of markov model hs codes which can be defined as follows :

S = s1,s2,s3,……sN {where S Є U (Universal Set) where N = 17}

T = t11, t12, t13, …………. tN1,………tNM {where T Є transition probability matrix, where each tij represent the probability of state i to transit to state j.

Π = π1, π2, ………..πN Π Є to an initial probability distribution over the state, πi , N = 17

This markov chain will further designed HMM λ= (T, €,π) for hs codes and will be defined as following:

T = t11, t12, t13, …………. tN1,… tNM

A transition probability matrix T, where each tij represent the probability of state i to transit to state j. Summation of each row and column of the transition matrix should be 1; N=M = 17

€ = €i(ϕt) Emission probability of observation being generated at state i

Φ = ϕ1, ϕ2 ………ϕT Set of T observation from the set of possible observation at a state

S = s1,s2,s3,……sN a set on N state; here N = 17

Π = π1, π2, ………..πN an initial probability distribution over the state, πi is the probability that the markov chain will start at state i and if πj = 0 then state j can be never be initial state; here N = 17

This defined HMM λ= (T, €,π) will be formulating the mathematical base for the derivation of the h_s code and finally the Hy–S indexer.

5.3 INFERENCE

This chapter present the novel location code called as "h_s code". The chapter explain the creation, traversal and mathematical model for the h_s encoder. This unique location code will become the base for the working of CBDFI search algorithm.

CHAPTER 6

CBDFI-SPATIAL SEARCH ALGORITHM

To search spatial data using CBDFI search algorithm, an efficient indexer is required. This indexer should be able to produce fast result keeping a desirable trade off with the operation cost and speed and load on system and skew time. The encoding of the sexagesimal location into h_s code and future into hexadecimal code will help in storing the location in consolidate and easily retractable format. But for the efficient search on the location and the associated keywords a hybrid indexer need to be get generated. In any spatial query related search engine two different indexer are used for extracting textual keyword and for location. The variant of inverted index used in multiple reference work for searching keyword in the spatial query and concept of minimum bounded rectangle is used for spatial indexing using quad-tree, kd-tree or r-tree. In order to improve efficiency of the search the order of calling the indexer is decider i.e. if location indexer first or textual indexer first and their variant presents different result under different condition. The proposed model will used a similar indexing technique for both textual and "hs code" and will generate the quad tree represent both the keyword and location relevance to search spatially.

6.1 HYBRID HY-S INDEXER

Both the inverted index and hash table are the two most used among the various different indexer used to process textual data. The inverted index yield better result while working you large dataset whereas the hash table need to be moderation with increasing data size. But if the size of the hash table can be identified before the formation of the hash function the result yield better performance with any of the conventional indexer. Therefore in this CBDFI model a hybrid indexer is generated with utilized the benefit of both the inverted index and hash table i.e. a combine indexer is generated which used both hash table first and then inverted index to fill the keyword in it dictionary.

The hybrid indexer of CBDFI model will yield the following result

Index number 1

Keyword1 → doc1.name(occurrences no.) → doc2.name(occurrences no.)

Keyword2 → doc1.name(occurrences no.) → doc2.name(occurrences no.)

And so on

Index number 2

Keyword m → doc1.name(occurrences no.) → doc2.name(occurrences no.)

Keyword m+1 → doc1.name(occurrences no.) → doc2.name(occurrences no.) And so on

.

.

Index number (key-1)

Keyword n → doc1.name(occurrences no.) → doc2.name(occurrences no.)

Keyword n+1 → doc1.name(occurrences no.) → doc2.name(occurrences no.) And so on.

Hence the textual keyword as well as h_s code will be indexed in the same fashion and thereafter will get converted into quad-tree. The pseudocode for both the inverted index and hashing is as follows

Algorithm 6-1 hybrid Inverted Index

```
1. Begin
2. Declare a macros MAX ← 500.
3. Define a structure INDEXING.
4. In struct indexing, declare character arrays NAME, FNAME1, FNAME2; two integers OCC1
   and OCC2.
5. Declare main.
6. Declare a struct indexing pointer INDEXX to store later on, all the keywords in NAME, their
   occurrence and count in FNAME1(OCC1) and FNAME2(OCC2) as per documents.
```

```
7.  Dynamically assign memory to INDEXX, index← malloc(100000*sizeof(struct
    indexing)).
8.  Initialize 3 file pointers fp1, fp2 and fp3.
9.  Declare two character pointers fileText1 and fileText2 to store files as strings.
10. Dynamically assign memory to both character pointers, fileText1 and fileText2←
    malloc(100000*sizeof(char)).
11. Initialize different integer variables (i, j, k, n, m, count, c11, c2, p) 0.
12. Initialize delimiter delim← " " .
13. Declare two character double pointers word and word2 to store the words from the
    strings.
14. Dynamically assign memory to both character double pointers, word and word2←
    malloc(70000*sizeof(char)).
15. Declare a character pointer STOPWORDS with limit MAX and initialize all the stopwords(e.g.
    a, an, he, she etc.) to it that needs to be removed without punctuations.
16. FP1←fopen(path to 1st file, read mode).
17. FP2←fopen(path to 2nd file, read mode).
18. FP1←fopen(path to new file in which keyword and their count would be saved, write
    mode).
19. If FP1 OR FP2 equals NULL
        a. Print "cannot open file" message.

        b. exit(0) to end the code immediately.
20. End if.
21. While FP1 doesn't reaches end of file; (!feof(fp1))
        a. fileText1[i]←fgetc(fp1) i.e. get each character from fp1 and write it to
           fileText1.
        b. Increment i.
22. End while.
23. I←0.
24. Repeat steps 21 to 23 for FP2 and fileText2[i].
25. word[i]← strtok(fileText1, delim) to tokenize 1st word from the string on delimiter basis.
26. While word[i] doesn't reaches NULL
        a. Increment i.
        b. word[i]← strtok(NULL, delim) Syntax for strtok function.
27. End while.
28. I←0.
29. Repeat steps 25 to 28 for word2[i] and fileText2 with same delimiter.
```

```
30. While word[i] doesn’t reaches NULL.
    a. For loop j← 0 to length of word[i]
        i. Word[i][j]← tolower(Word[i][j]) converting each letter to lower case.
        ii. While word[i][j] i.e. the letter doesn’t belong to (a to z) or (0 to 9) or
            NULL.
            i. If word is ‘\n’ i.e. an Enter or change of line, replace it with ‘\0’
               i.e. NULL.
            ii. For loop k← j to length of word[i]
                a. word[i][k]← word[i][k+1] i.e. replace it with next
                   character and so on to remove that punctuator from
                   whole word.
            iii. End for loop.
        iii. Word[i][k]← ‘\0’ to End while.
    b. End for loop.
    c. While stopwords[n] doesn’t reaches NULL.
        i. If strcmp(word[i],stopwords[n]) i.e. word is equal to any of the
           stopwords.
            i. Increment the count.
            ii. Break the while loop.
        ii. End if.
        iii. Increment n.
    d. End while.
    e. If count equals 0 i.e. the word[i] is a keyword.
        i. If c11 is not equal to 0.
            i. P← 0.
            ii. While p is less than c11.
                a. If word[i] equals any of the previous index[p].name i.e. to
                   count number of occurences.
                    i. Increment index[p].occ1.
                    ii. Increment c2.
                b. End if.
                c. Increment p.
            iii. End while.
        ii. End if.
        iii. If c2 equals 0.
            i. Increment c11.
            ii. Copy word[i] content to index[m].name using strcpy() function.
```

```
                iii. index[m].occ1← 1 since it is a new word.
                iv. Copy the name of the first document to index[m].fname1.
                v. Increment m.
            iv. End if.
        f. End if.
        g. Count← 0.
        h. Increment i.
        i. (n and c2)← 0.
31. End while.
32. (I, n, count, c2, p)← 0.
33. While word2[i] doesn't reaches NULL.
        a. For loop j← 0 to length of word2[i]
            i. Word2[i][j]← tolower(Word2[i][j]) converting each letter to lower case.
            ii. While word2[i][j] i.e. the letter doesn't belong to (a to z) or (0 to 9) or
                NULL.
                i. If word is '\n' i.e. an Enter or change of line, replace it with '\0'
                   i.e. NULL.
                ii. For loop k← j to length of word2[i]
                    a. word2[i][k]←word2[i][k+1] i.e. replace it with next
                       character and so on to remove that punctuator from
                       whole word.
                iii. End for loop.
            iii. Word2[i][k]← '\0' to End while.
        b. End for loop.
        c. While stopwords[n] doesn't reaches NULL.
            i. If strcmp(word2[i],stopwords[n]) i.e. word is equal to any of the
               stopwords.
                i. Increment the count.
                ii. Break the while loop.
            ii. End if.
            iii. Increment n.
        d. End while.
        e. If count equals 0 i.e. the word2[i] is a keyword.
            i. If c11 is not equal to 0.
                i. P← 0.
                ii. While p is less than c11.
                    a. If word2[i] equals any of the previous index[p].name i.e. to
                    count number of occurences in 2nd document.
                        i. Increment index[p].occ2.
                       ii. Copy the name of the 2nd document to
                           index[p].fname2.
                      iii. Increment c2.
                    b. End if.
                    c. Increment p.
               iii. End while.
          ii. End if.
         iii. If c2 equals 0.
               i. Increment c11.
              ii. Copy word2[i] content to index[m].name using strcpy()
                  function.
             iii. index[m].occ2 ← 1 since it is a new word.
              iv. Copy the name of the second document to index[m].fname2.
               v. Increment m.
          iv. End if.
     f. End if.
     g. Count ← 0.
     h. Increment i.
     i. (n and c2) ← 0.
34. EndWhile.
35. I ← 0.
36. While p is less than c11 where c11 contains total number of keywords.
     a. Write string index[p].name to fp3 using fprintf().
     b. If index[p].occ1 is not equal to 0.
           i. Write string index[p].fname1 as "->docname1" to fp3.
          ii. Write count index[p].occ1 as "(count)" to fp3.
         iii. If index[p].occ2 is not equal to 0.
               i. Write string index[p].fname2 as "->docname2" to fp3.
              ii. Write count index[p].occ2 as "(count)" to fp3.
          iv. End if.
           v. Write "\n" to fp3 to change the line.
     c. End if.
     d. Else If index[p].occ2 is not equal to 0.
               i. Write string index[p].fname2 as "->docname2" to fp3.
              ii. Write count index[p].occ2 as "(count)" to fp3.
             iii. Write "\n" to fp3 to change the line.
```

```
        e. End else if.
        f. Increment p.
37. End while.
38. system(" time ./a.out") to print total time taken.
39. fclose fp1, fp2 and fp3.
40. End.
```

The second algorithms that define the hash key is described underneath

---

Algorithm 6-2 Hybrid Hash table

```
1. Begin.
2. Declare a macros MAX← 500.
3. Define a structure Node.
4. In struct Node, declare character array DATA and a struct node pointer NEXT.
5. Declare a function append(struct node** head_reference, New Data)
    a. Allocate new_node using malloc.
    b. Last←head_reference
    c. Copy New Data to new_node->data.
    d. Assign NULL to new_node->next.
    e. If head_reference is NULL, then
        i. Head_reference← new_node.
        ii. End the function.
    f. End if.
    g. While last->next doesn't reaches NULL i.e. traverse through list
        i. Last← last->next.
    h. End while.
    i. Last->next← new_node.
6. End function declaration.
7. Declare a function printList(struct Node *node, FILE *fp)
    a. C← 0.
    b. While node doesn't reaches NULL
        i. Write to the file fp, node->data as "data\n" using fprintf().
        ii. Increment c.
        iii. Node← node->next.
        c.  End while.
        d.  Print the number of words i.e. c.
8.  End function declaration.
9.  Declare main.
10. Initialize 2 file pointers fp1 and fp2.
11. Initialize hash_size as 47.
12. Declare a struct node pointer array head of size hash_size.
13. Declare a character pointers fileText1 to store file content as a string.
14. Dynamically assign memory to the character pointer, fileText1 ←
    malloc(100000*sizeof(char)).
15. Initialize different integer variables (i, j, k, n, m, count, p, key)← 0.
16. Initialize delimiter delim← " " .
17. Declare two character double pointers word and new to store the words and then
    keywords respectively from the string.
18. Dynamically assign memory to both character double pointers, word and new←
    malloc(70000*sizeof(char)).
19. Declare a character pointer STOPWORDS with limit MAX and initialize all the
    stopwords(e.g. a, an, he, she etc.) to it that needs to be removed without punctuations.
20. FP1←fopen(path to 1st file, read mode).
21. FP2←fopen(path to new file in which keywords at their respective indexes would be saved,
    write mode).
22. If FP1 equals NULL
        a.  Print "cannot open file" message.
        b.  exit(0) to end the code immediately.
23. End if.
24. While FP1 doesn't reaches end of file; (!feof(fp1))
        a.  fileText1[i]← fgetc(fp1) i.e. get each character from fp1 and write it to
            fileText1.
        b.  Increment i.
25. EndWhile.
26. I← 0.
27. word[i]← strtok(fileText1, delim) to tokenize 1st word from the string on delimiter basis.
28. While word[i] doesn't reaches NULL
        a.  Increment i.
        b.  word[i]← strtok(NULL, delim) Syntax for strtok function.
29. Endwhile
30. I←0 .
31. While word[i] doesn't reaches NULL.
        a.  For loop j← 0 to length of word[i]
```

```
            i. Word[i][j]← tolower(Word[i][j]) converting each letter to lower case.
           ii. While word[i][j] i.e. the letter doesn’t belong to (a to z) or (0 to 9) or
               NULL.
                   1. If word is ‘\n’ i.e. an Enter or change of line, replace it with ‘\0’
                      i.e. NULL.
                   2. For loop k← j to length of word[i]
                          a. word[i][k]← word[i][k+1] i.e. replace it with next
                             character and so on to remove that punctuator from
                             whole word.
                   3. End for loop.
          iii. Word[i][k]← ‘\0’ to End while.
      b. End for loop.
      c. While stopwords[n] doesn’t reaches NULL.
            i. If strcmp(word[i],stopwords[n]) i.e. word is equal to any of the
               stopwords.
                   1. Increment the count.
                   2. Break the while loop.
           ii. End if.
          iii. Increment n.
      d. End while.
      e. If count equals 0 i.e. the word[i] is a keyword.
            i. new[m]← strdup(word[i]) i.e. makes a duplicate copy of word[i] and saves
               it in new[m].
           ii. Increment m.
      f. End if.
      g. (count and n)← 0.
      h. Increment i.
32. End while.
33. For p← 0 to hash_size, do
      a. Head[p]←NULL.
34. End for.
35. While new[m] is not equal to NULL
      a. Len← length of new[m].
      b. Key← sum of integer ASCII equivalents for first, middle and last characters of
         new[m].
      c. Key← key%hash_size.
      d. Call append function as append(&head[key],new[m]).
```

```
        e. Increment m.
36. End while.
37. For i← 0 to hash_size
        a. Print meassage “At index (i)”.
        b. Write the same message to fp2.
        c. Call the function printList(head[i],fp2).
38. End for.
39. system(“ time ./a.out”) to print total time taken.
40. Fclose fp1,fp2 and fp3.
41. End
```

6.2 CBDFI-SPATIAL SEARCH MODEL

Spatial keyword search index is incomplete until but the keyword and spatial location are indexed. Most of the research perform these both task separately but for more efficient result a pruning of keywords and the text should be performed at same time and with same indexing technique. Use of data driven spatial index are not scalable and their support is expensive hence the proposed index use space driven index i.e. quad-tree for better results

Hy-S Indexer perform textual indexing first where from the provided document Ɖs (spatial, attribute based or both) the keyword dictionary (φ) is generated with each keyword (k) is given a weight (ψ) depending upon its relevance and importance in the context. A threshold value is used to identify whether the given keyword will be useful for the operation or not. Hy-S indexer used quad-tree from spatial indexing because of its low cost while maintain the data structure and its uniform distribution mechanism.

Hy-S indexer will be having excessing keyword from the dictionary, followed by checking it relevance in the quad-tree and finally storing the information in the inverted index file. The indexed number will be decided by the hash table. The number of the keyword in the dictionary and the number of matched location in the quad-tree will help us in identify the hash function and further the hash key that will store the inverted index file with then. The spatial relevance and the textual relevance is calculated to match the keyword in the search with the keyword in the dictionary and if a match is registered that that location node is pointed which is associated with different location that can be addressed that can have the similar keywords in them.

The spatial model of CBDFI model can be described by following formulas and the respective notations are used (table 6-1)

Table 6-1Notation of CBDFI search Model

Spatial Object	Ob_s
Object Location	Ob_s.loc
Keywords	ķ
Query	∂
Query Keywords	∂. ķ
Dictionary	φ
Dictionary of keywords	φ_{key}
Dictionary of h_s code	φ_{hs}
Weight	ψ
Weight of keywords	ψ_{key}
Weight of h_s code	ψ_{hs}
H_s code	h_s
Document	Đ
Ranking function	Γ
Length of keyword dictionary	l_k
Length of spatial dictionary	l_h

The spatial search indexer is based on the inverted index, hash table and quad-tree. Along with the same the search engine also utilize the hidden markov model of them for calculating the probability of transmission of h_s code and calculation of their weight to travel from one code to another, this weight decide the distance between the two nodes of the h_s quadtree which is whole responsible for this search algorithm. Various different formulation used in this research with their explanation is described underneath.

Hash function: this function is responsible for finding out the size of the hash table and is calculated as follows

$$hash\,key = \sum_{i=1}^{dict\ length} \psi(\mathrm{i}) * \Gamma/m$$

Where the length of the dictionary could be l_k for keyword and l_h for spatial objects, ψ weight of the key or hs code and Γ is the ranking function defined to express the tectual and spatial combine impact factor. And m is defined as the length of dictionary modulus p Є palpha{}.

$$m = l\_h\%p$$

The ranking function is defined as the summation of the spatial relevance, textual relevance, hidden markov model transition and emission probability and the hs distance calculation

Any query **(**∂**)** fired in the CBDFI model will be defined as (∂:loc; ∂:doc; k). where ∂:loc is the spatial representation of lat/long pair encoded in h_s code, ∂:doc is the set of keywords in the query ∂ and k is the parameter in the Top k query. A ranking function (Γ) will compute the relevance score of spatial object D_i and the query ∂

$$\Gamma_{total} = \Phi\,\Gamma_{sp} + (1- \Phi)\,\Gamma_{tx}$$

Where Γ_{sp} is the spatial ranking score and Γ_{tx} is textual ranking score and Φ Є {0- 1} defines the relevance, i.e. not all query will have same importance of location and keyword some may have more preference over others. Spatial ranking score will be calculated as follows:

$$\Gamma\mathrm{sp}\ (\mathrm{Di{:}\ doc};\ \boldsymbol{\partial}{:}\ \mathrm{doc}) = 1 - \frac{\mathrm{Dis(Di{:}\ loc};\ \boldsymbol{\partial}{:}\ \mathrm{loc})}{Max\_dis}$$

And the textual ranking score will be calculated depending on various factors.

$$\Gamma tx\ (Di{:}\,doc;\ \boldsymbol{\partial}{:}\,doc) = \sum\nolimits_{t\,\in Di:doc \cap\, \boldsymbol{\partial}:doc} \left(\frac{f(t, Di{:}\,doc)}{|Di{:}\,doc|} * \log\left(\frac{|\varphi key|}{|\{Di{:}\,doc\ \in \varphi key | t \in Di{:}\,doc\}|}\right) * \frac{f(t, \boldsymbol{\partial}{:}\,doc)}{|\boldsymbol{\partial}{:}\,doc|} * \log\left(\frac{|\varphi key|}{|\{\ \boldsymbol{\partial}{:}\,doc\ \in \varphi key | t\ \in\ \boldsymbol{\partial}{:}\,doc\}|}\right)\right)$$

$$/ \sum_{t=1}^{|Di:doc|} \sqrt{\frac{f(t, Di{:}\,doc)}{|Di{:}\,doc|} * \log\left(\frac{|\varphi key|}{|\{Di{:}\,doc\ \in \varphi key | t \in Di{:}\,doc\}|}\right)^2}$$

$$* \sum_{t=1}^{|\boldsymbol{\partial}:doc|} \sqrt{\frac{f(t, \boldsymbol{\partial}{:}\,doc)}{|\boldsymbol{\partial}{:}\,doc|} * \log\left(\frac{|\varphi key|}{|\{\ \boldsymbol{\partial}{:}\,doc\ \in \varphi key | t\ \in\ \boldsymbol{\partial}{:}\,doc\}|}\right)^2}$$

The hybrid search model used in CBDFI model Top k spatial keyword search query to fetch the result from the column database store in form of a spatial quad-tree designed specifically for most relevant and frequent keyword processed by the system. System is designed in such a way that it will render itself as per the data feed and present most optimal result when used to search the same family keywords. Another important aspect of this model is that the node of independent quad-tree can address 24 distinct location simultaneously in a single cycle and all the required location and their corresponding keywords will be stored in same index at same level and under the singular node. The ranking function defined above will be render with the firing of the spatial query where the associated query parameter with the "h_s code" is generated which identify required location for the search.

6.3 Hi-S TREE

The formation of the Hs-I tree plays the most important role in the fast running of the CBDFI model. The every "h_s code" encounter by the system the quad tree start taking shape. The calculated probability of "h_s code" to traverse from one "h_s code" to another decide the formation of the quad tree. Along the present "h_s code" the most frequent "h_s code" is identified and the distance of each "h_s code" from most frequent "h_s code" is calculated and registered. On the base of the most frequent "h_s code" and their respective distance the limit of the quad-tree and the insertion of the new element in the quad-tree is decided.

Hs-I tree of the CBDFI model will be stored in files on the memory with predefined size. For each spatial object Ob_s a fixed size slot will be fixed in the file page. The number of the spatial object in the page will be decided as per the ration of the page size and bit size of spatial object i.e. 7 bit. Similar h_s code will be stored in the same page under the same node of the quad tree. The h_s code will be tagged with id with respect to the maximum frequency keyword to ease the transition of the h_s code from the memory

Let us assume that first hs1 code is registered by the system with probability prob(h_s (1)) be x, so this node will be considered as the parent node. On arrival on the next h_s (2) code, the probability prob(h_s (2)) is calculated and let it be y. Hence if the probability y>x then the h_s (2) will replace h_s (1) and will become the new parent node and h_s (1) will become the child node of the tree. On the arrival of the next hs(3) code, the same process will be repeated i.e. the probability of h_s (3) will be calculated and compared with the present "h_s codes" and if its probability is the highest then h_s (30 will replace the parent node. The process will keep on till all the four node of the quadtree is achieved. Along with the calculation of probability also the distance and averaging of the distance is taken care. If the distance of the upcoming "h_s code" is not similar to that level child node then that "h_s code" will be added as the child node in the next level to the parent which have the least distance and within the limit of tree.

6.3.1 INSERTION IN Hs-I TREE

In the process of insertion of new node in the hs-I quad tree location table is checked. If the location table is empty or the no reference of the "h_s code" is identified then empty slot in the file f is identified and "h_s code" is stored there. With each write operation on the file, the location table is updated. Along with the calculation of the probability of traversal of the "h_s code" is also generated in stored in the location probability table and maximum probability pointer is also updated If the "h_s code" in present in the location table, the probability of that "h_s code" is calculated and compared with the maximum probability and swapped with latter in case of higher probability.

The following algorithms will explain the insertion deletion and updating of the hs-I

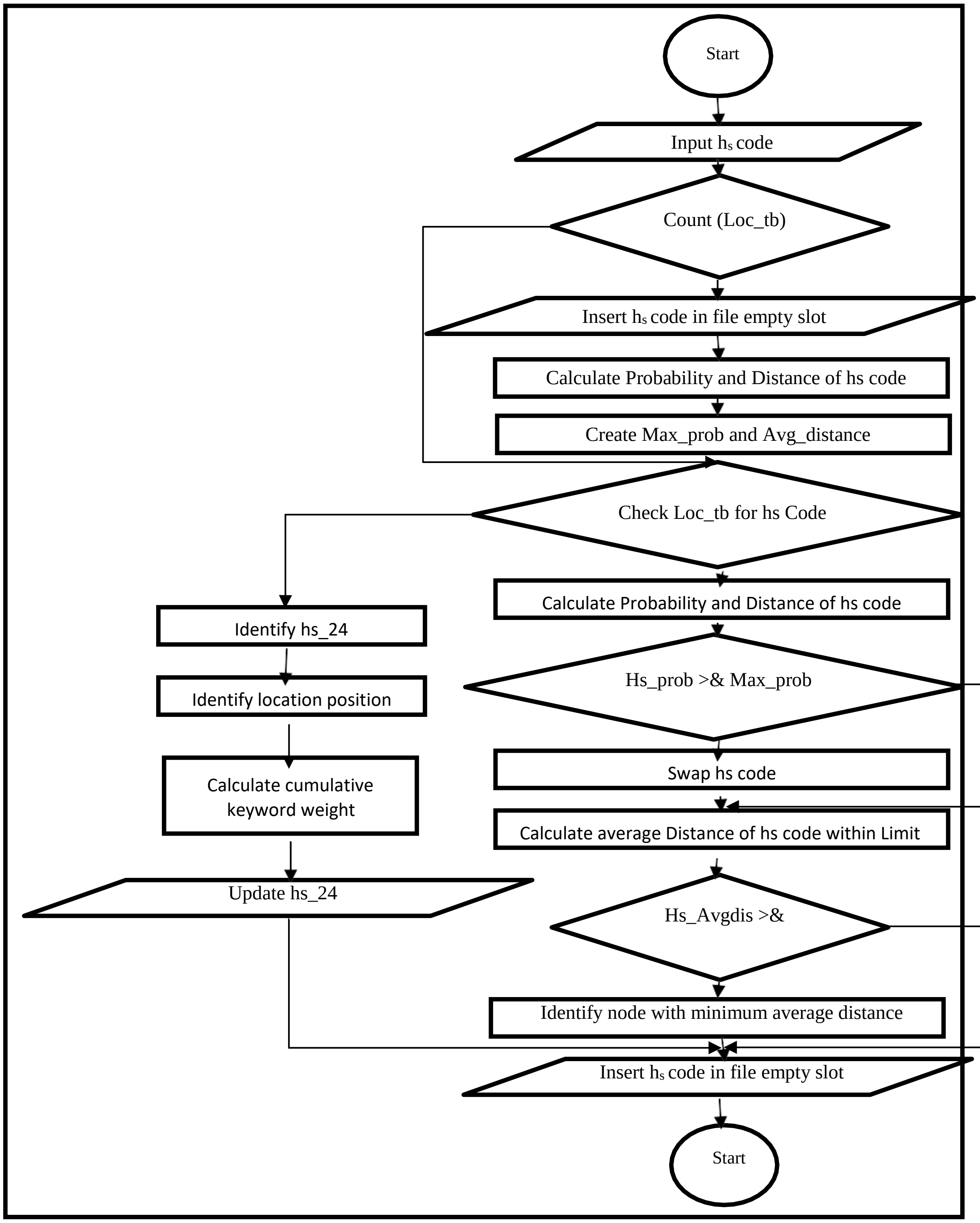

Start
Input hs code
Count (Loc_tb)
Insert hs code in file empty slot
Calculate Probability and Distance of hs code
Create Max_prob and Avg_distance
Check Loc_tb for hs Code
Calculate Probability and Distance of hs code
Identify hs_24
Hs_prob >& Max_prob
Identify location position
Swap hs code
Calculate cumulative keyword weight
Calculate average Distance of hs code within Limit
Update hs_24
Hs_Avgdis >&
Identify node with minimum average distance
Insert hs code in file empty slot
Start

Fig6-1: Process flow for inserting node in Hs-I tree

Algorithm 6-3 Insert in Hi-s tree

```
1: hs_code ← input()
2: if loc_hs is null
3:         f_hs ← open("locationhs.txt","w")
4:         f_hs.write(hs_code)
5:         prob_hs← Cal_probability('hs_code')
6:         mx_hs ← hs_code
7:         dis_hs← Cal_distance('hs_code')
8:         max_prob← prob_hs
9:         avg_distance← $avg(dis_hs)
10: else:
11: for i in range(size(loc_hs)
12: if (hs_code == loc_hs(i))
13:         f_hs←1
14:         else:
15:                 f_hs←0
16:if (f_hs == 0)
17:         prob_hs← Cal_probability('hs_code')
18:         dis_hs← Cal_distance('hs_code')
19:         if (prob_hs > max_prob)
20:                 temp← mx_hs
21:                 mx_hs← hs_code
22:                 hs_cur← hs_code
```

```
23:         hs_code← temp
24:                 avg_distance+1← $avg(dis_hs)
25:                 if (avg_distance +1 > avg_distance)
26:                         for i in range (size(loc_hs)
27                                  p_node ←min_dis(hs_cur, loc_hs(i))
28: else:
29: for i in range(size(hs_24)
30: hs_code = hs_24(i)
31: ind←i
32: for i in range(24)
32:         hs_code (location) == hs_24(ind).location
33: calculate_cum_key_weight(hs_24(ind),key)
34: f_hs.write(hs_code)
35: f_hs.close()
```

6.3.2 DELETION AND UPDATION IN Hs-I TREE

The deletion operation will be performed on the Hs-I tree whenever there will be a need for updating the nodes of the h_s-24 node. On identification of the new location code or keyword the CBDFI model insert the new node in the hs- I tree and update the weight median values of the keyword at that hs-24 node. Therefore to update weight median of keyword the target hs-24 node will be identified, and using the depth first techniques the left most child node will be checked for the suitable projected distance. All 6 location weight median values will be deleted and inserted again with the updated value.

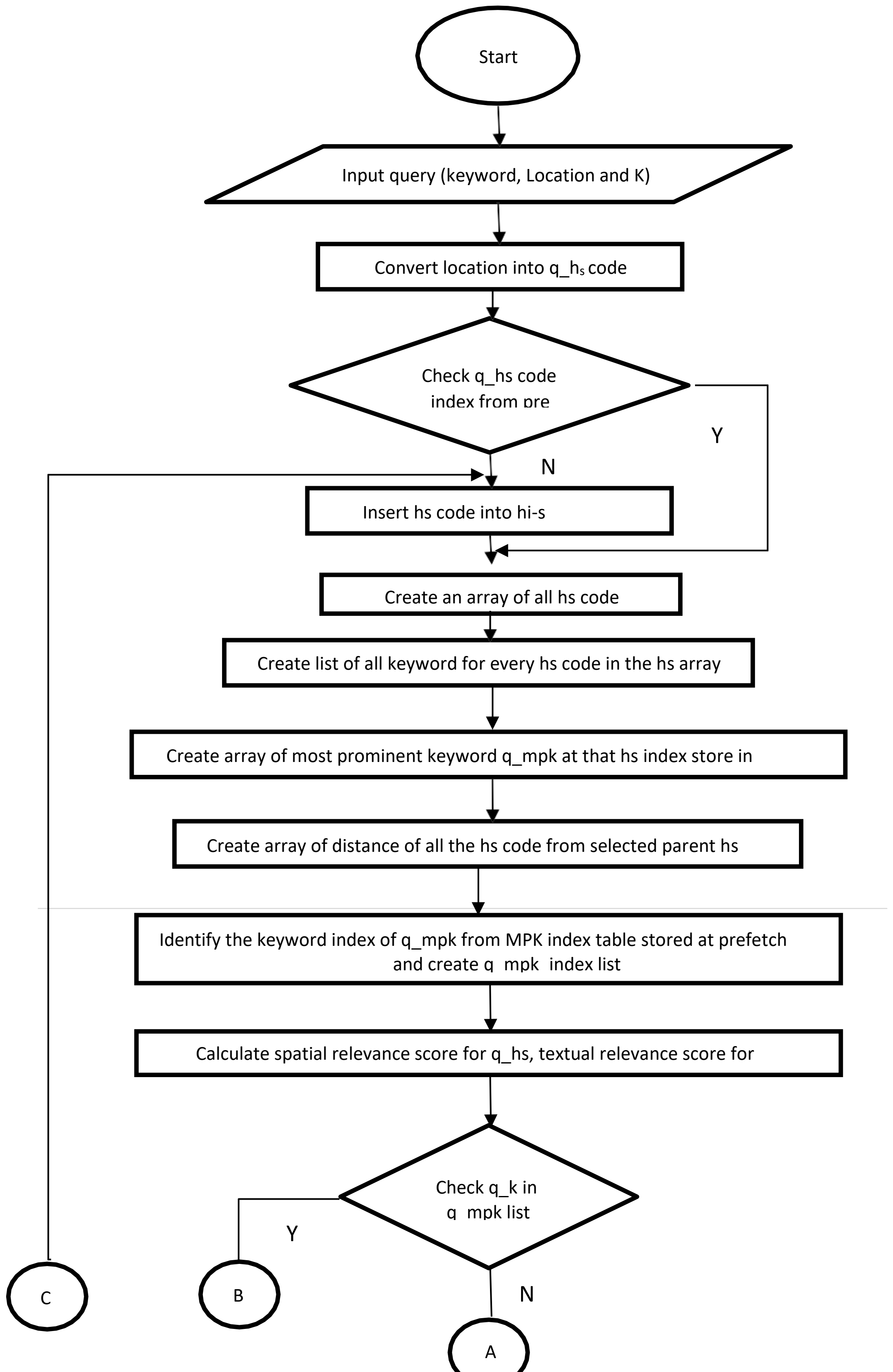
6.4 CBDFI- SEARCH ALGORITHM

The CBDFI model, search spatial object by traversing through the nodes of Hi- S tree. Hybrid spatial indexing algorithm, index keywords associated with a particular hs-24 and store them at the nodes of the quad-tree. For searching any spatial object, the respective "h_s code" node will be identified and all location associated with that "h_s code" or that "h_s code" index will be searched for positive match of the keyword. Hence this model eliminate remaining hs-24 node which are not associated with query location and optimize the number of location to be searched. The Query optimizer designed for the CBDFI model, utilize a pre-fetch table to store list of various location h_s code and global probable keyword at respective "h_s code". Pre fetch table plays most crucial role in reducing the fetch time of the query and improving the execution time of the query itself.

The CBDFI model extract query keywords and location from top k spatial keyword query and search spatial object with similar or correlated keyword. Search algorithm convert the query location into "h_s code" and check for the same in the list of "h_s code" in the pre-fetch table either the "h_s code" itself or for the index of "h_s code". If user location is new then the generated "h_s code" is inserted in the Hs-I Tree and recalibrate the pre-fetch table. Therefore the search algorithm check whether the user location's index can be identified and all the associated location will that index will the subset of the location where the required keyword need to be matched.

The CBDFI model process each and every keyword extracted for the dataset and weight them on following bases: 1) location of a keyword, 2) correlation of the keywords with the nearby keyword, and 3) probability of occurrence of keyword at that location. Each location in hs -24 node will have multiple keywords associated with one or many spatial object at that location. Therefore; from a list of various keywords and their frequency of repetition for a particular spatial object, the most frequent keyword at that location will be identified. In case of having multiple spatial objects at the same location, most frequent keyword of each object will be correlated with all the keywords at that location and based on heuristic study the probability of occurrence of a particular keyword will be calculated The local probable keywords for that location will be identified based on these probabilities. As multiple location comprise of the particular h_s -24 node, hence local probable keywords for h_s -24 will be identified. Further multiple hs-24 code are linked under same index hence the global probable keywords for that index will be identified and stored in the pre-fetch table. Hence the CBDFI model's pre-fetch table turn out to be the first location where the query keyword is checked.

The CBDFI search model will follow three search routines to identify spatial object with the query keyword. In the first search routine, search algorithm will read the list of global probable keywords associated with the index of the user "h_s code" and will identify user keyword. On a positive identification of a keyword, top k location with minimum distance score from the user location will be returned back to the user as the query result. The CBDFI search algorithm will return positive match with the time complexity of O(1) for directed associated keywords. To identify indirectly associated keywords the CBDFI search algorithm will follow the second search routine where the model will retrieve the list of all the local probable keywords associated with that "h_s code" index. Form this list of keywords the user keyword will be matched and location of the spatial objects will be identified. This is the divide and conquer approached followed by the CBDFI model where the only identified index of "h_s code" will be searched for the result of the query. And in the worst case the third routine will be executed where the whole Hs-I tree will be searched for user keyword and the result will be populated back to the user and pre-fetch table will be updated. Hence the CBDFI model search directed associated as well as indirectly associated keyword for the user query. The system is designed in such a way that with every search query, the system update the pre-fetch table with new location and keyword. With more number of queries better and more efficient the search algorithm run and it cater all the new keyword and location to populate updated index and Hs-I tree. The following flowchart will explain the working of the search algorithm in figure 6-2.



Start
Input query (keyword, Location and K)
Convert location into q_hs code
Check q_hs code index from pre
Y
N
Insert hs code into hi-s
Create an array of all hs code
Create list of all keyword for every hs code in the hs array
Create array of most prominent keyword q_mpk at that hs index store in
Create array of distance of all the hs code from selected parent hs
Identify the keyword index of q_mpk from MPK index table stored at prefetch and create q_mpk index list
Calculate spatial relevance score for q_hs, textual relevance score for
Check q_k in q_mpk list
Y
N
C
B
A

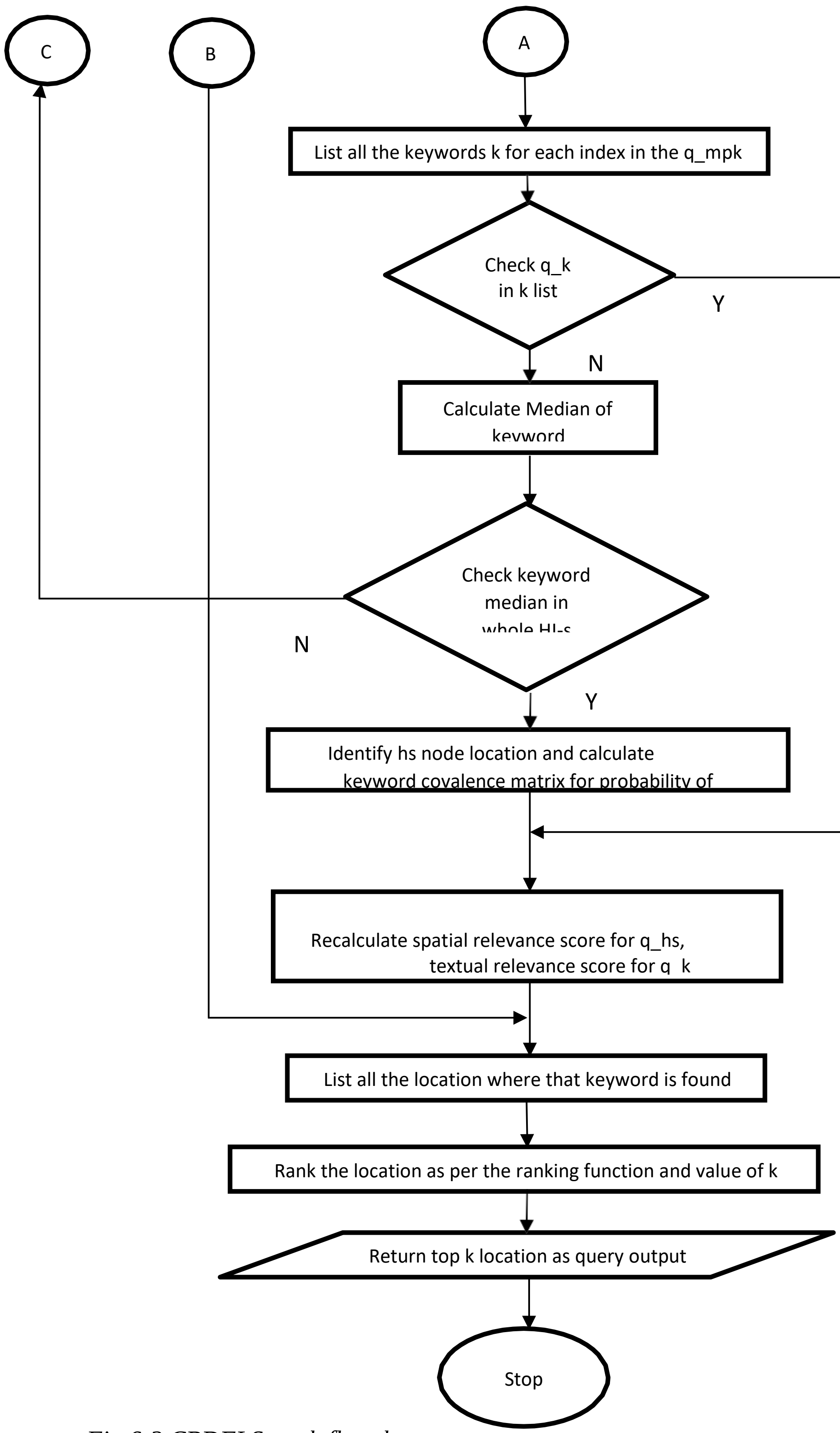

C
B
A
List all the keywords k for each index in the q_mpk
Check q_k in k list
Y
N
Calculate Median of keyword
Calculate cumulative probability of keyword
Check keyword median in whole HI-s
N
Y
Identify hs node location and calculate keyword covalence matrix for probability of
Recalculate spatial relevance score for q_hs, textual relevance score for q_k
List all the location where that keyword is found
Rank the location as per the ranking function and value of k
Return top k location as query output
Stop

Fig 6-2 CBDFI Search flowchart

The algorithm of CBDFI search algorithm is as follows:

Algorithm 6-4: Search algorithm for CBDFI model

```
1: q_loc🡐 ∂.location
2: q_key🡐 ∂.keyword
3: q_k🡐 ∂.k
4:  q_hs🡐 generate_hs(q_loc)
5:   for i in range each pf_hs[]
6:              if( q_hs != pf_hs[i])
7:                          Insert_hi_stree(q_hs)
8:              else:
9:              for i in range size(hs_index[])
10:                         Ar_hs[]= hs_index[i]
11:                         for j in range size( hs_index[i].keywords[])
12:                                     Ar_hs_key_list[] = hs_index[i].keywords[j])
13:                                     q_mpk[]🡐
     calculate_mostprominetkeyword(Ar_hs_key_list[])
14:             Cal_spatial_rel(q_hs)
15:             Cal_textual_rel(q_hs)
16: For I in range size(q_mpk[])
17:             If (q_key != q_mpk[i])
18:                         For j in range(q_mpk[i].index[])
19:                                     Q_mprk[] =q_mpk[i].index[j]
20:             Elseif(q_key != q_mprk[])
21:                         Q_key_mediam🡐 calculate_median(q_key)
22:                         For I in range size(hi_s_key_median[])
23:                                     If (q_key_median == hi_s_key_median[i])
24:                                                 m_hs🡐Identify_hs_loc(q_key)
25:                                                 Loc_match🡐1
26:                         If (loc_match !=1)
27:                         Goto step 7:
28:             Else:
```

```
29:         Cal_cum_prob(q_key)
30:         Cal_spatial_rel(m_hs)
31:         Cal_textual_rel(m_hs}
32:         Match_loc[]←hs_q_mpk[i]
33: Calculate_rank(q_k)
34: R_loc← ∂.result
```

6.4.1 ACCURACY

The CBDFI model, search spatial object by traversing through the nodes of Hi-S tree. For any spatial object, their location "h_s code" must reside on the node of the Hi-S tree. If the object location have some associated keywords which have high probability of occurrence than that location keyword will be present in the node of the quad tree. Therefore identification of hs-24 node in the Hs-I tree is done by calculating the projected_distance of the query location "h_s code" and identify projected_distance2 area of the hs- 24 node in Hs-I tree where user's projected_distance lies in the range. This projected_distance2 will act as the splitting function of the node of the Hs-I tree and the construction of quad tree ensure that division of the node will be based on the projected_distance2 of the "h_s codes". Hence the CBDFI model correctly identify the exact hs_24 node where the location will be present and there will be no other node in the Hs-I tree where that location can be allocated. Now each location of the hs-24 code will have median of weight of the local probable keywords and if the query keyword weight doesn't change the median of that node then that location will be correct location where the required location of the spatial object of the query reside. Hence validating the correctness of the CBDFI search algorithm.

6.4.2 TIME COMPLEXITY

The CBDFI model, search spatial object by traversing through the nodes of Hi-S tree let us assume that there are l_h spatial object in the spatial dictionary and l_k keywords in the keyword dictionary designed within the search region of the algorithm. The indexing algorithm designed for the CBDFI model will divide, various hs code into m groups of spatial objects. Therefor the time complexity for all the three subset will be calculated. For searching any object in the list (size n) the time complicity will be O(n) which will be best case scenario for the CBDFI search algorithm. For search the second case when the list of local probable keywords for each spatial object in the search region, the worst case time complexity of search an object will be O(((l_h/m)-1)l_k) . And time complexity for searching a keyword in the list of keywords using a link list will be O(l_k). Therefor for the third case where all the element of the time complexity for the worst case seniors will be O(n(1+ l_k *((l_h –m)/m)l_k))). In the empirical study, the performance of the algorithm is highly efficient as a large number of objects and quad tree nodes are eliminated by making use of projected distance and weight median of the keywords

5.3 INFERENCE

This chapter present the novel location search algorithm based on h_s code and for the column database. The chapter also provide the time complexity of the proposed algorithm of O(n) in the best case scenario when the keyword will be in the pre fetch table.

CHAPTER 7

RESULTS & DISCUSSION

In order to test the performance of the spatial search algorithm, the CBDFI model is tested on an airplane dataset, provided by the aviation department of the USA. This dataset is freely available in text (tab delimited) format for research purpose and provide year wise finding of 2500 flights departed from New York and Washington DC to South America, Africa, Europe, United Kingdom and India. This dataset provide information regarding capacity of aircraft, duration of travel with source-destination station, staff information, passenger information with luggage report, passenger reviews and aircraft maintenance report etc.

The CBDFI search algorithm assumed that the spatial object provided in the dataset will be associated with location and description document Đ where Đ will define the characteristic of spatial object. The CBDFI spatial object can be defined as follows:

$Ob_s(1) \rightarrow \{ Ob_s.loc(1), ķ(1)\}$

The CBDFI model is designed to index unique keywords identified from the textual information extracted from the description document Đ. Hence any spatial object will be collection of location $Ob_s.loc(1)$ and associated keywords ķ(1) mentioned in the document Đ. The $Ob_s.loc(1)$ will be converted by in h_s code by h_s encoder and ķ() will be set of all the relevant keywords that can be extracted from the document Đ. Hence

$Ob_s(1) \rightarrow \{ h_s(1), ķ(1)\}$

$Ob_s(2) \rightarrow \{ h_s(2), ķ(2)\}$

.

$Ob_s(z) \rightarrow \{ h_s(z), ķ(z)\}$

Hence all the z different spatial object detected by the CBDFI model will be converted into "h_s code" and the set of different keyword associated with them. The CBDFI model use the column database Monet Db to store "h_s code" generated by the system, where each column of “h_s code” will comprise of their corresponding keyword and keyword frequency of occurrence.

h_s(1)
ķ_1(1)
ķ_1(2)
ķ_1(3)
ķ_1(4)
.
ķ_1(n)

h_s(1)
freq(ķ_1(1))
freq(ķ_1(2))
freq(ķ_1(3))
freq(ķ_1(4))
.
freq(ķ_1(n))

These h_s columns will assist the system in calculating the highest frequency keywords associated with each h_s code. The calculated frequencies of keyword will succor in understanding the importance of the keyword and there relevance with respect to a given location code. A list of highest frequency keywords will be store in the linked list in the pre-fetch table and will act as the first benchmark for the search algorithm.

During the pre-processing stage, the system identify spatial objects which could be either at same location or at same generated h_s code. For existing “h_s code”, no new entry need to be addressed in the database as well as in the pre-fetch table as this location will be part of the 24 location of a given h_s node. For multiple spatial objects at given location, covalence matrix of the different keywords will be generated to identify the similarities and cumulative weights of the keywords at that location. This covalence matrix will be generated considering textual relevance of the keyword with respect to each other and that location. After the pre-processing stage the query optimizer will calculate the probability_weight of the “h_s codes” and their probability of transition from an h_s code to another h_s code. The maximum probability_weight of the generated “h_s code” with the will be consider as the parent node for the hi-s quad tree. Based on the probability_weight, distance ^{2} and distance of generated “h_s code” the hi-s quad tree will be created. This quad tree will provide the spatial indexer of the location captured by the CBDFI model, and each node of tree will be future associated with keywords registered at that “h_s code”. The hs-I quad tree will also consider the most probable keyword for all the keywords linked with the same keyword index and also the most probable keyword for all keyword associated with the same h_s index.

Location	**Loc_hs**	**Loc_prob**	**Loc_Distance**
Ob_s.loc (1)	h_s(Ob_s.loc (1))	prob(h_s(1))	Dis(Max(hs) -h_s(1))
Ob_s.loc (2)	h_s(Ob_s.loc (2))	prob(h_s(2))	Dis(Max(hs) -h_s(2))
Ob_s.loc (3)	h_s(Ob_s.loc (3))	prob(h_s(3))	Dis(Max(hs) -h_s(3))
Ob_s.loc (4)	h_s(Ob_s.loc (4))	prob(h_s(5))	Dis(Max(hs) -h_s(5))
.	.	.	.
Ob_s.loc (m)	h_s(Ob_s.loc (m))	prob(h_s(m))	Dis(Max(hs) -h_s(m))

With the identification of all the keywords, locations and converting “h_s code”, CBDFI model will now index all the keywords and “h_s code” present in the database. Over here the dictionary of keyword and “h_s code” will be generated which will be used as the feed for the hi-s indexer. Keyword dictionary will be having all the keywords registered from the dataset and will be linked with their respective weights

$$\phi_{key} = \{ k(1)^* \ \psi_{key}(1),\ k(2)^* \ \psi_{key}(2), \ldots\ldots\ k(l\_k)^* \ \psi_{key}(l\_k)\}$$

And the location dictionary will be having all the hs codes

$$\phi_{hs} = \{ h_s (1)^* \ \psi_{hs}(1),\ h_s (2)^* \ \psi_{hs}(2), \ldots\ldots.\ h_s (l\_h)^* \ \psi_{hs}(l\_h)\}$$

For both the φ_{key} and φ_{hs} hash function will be generated which will help us in generating the hybrid index for both keywords and hs code.

$$\text{Hash key(keyword)} = (ASCII(first) + ASCII(middle) + ASCII(last))/l\_k\%p$$

$$\text{Hash key(spatial)} = (ASCII(6) + ASCII(4) + ASCII(2))/l\_s\%p$$

where p Є p alpha{}, i.e. the denominator should be prime factor and for the hs code. From this hash function following index will be generated:

Index keyword

Index no 0

Keyword1→(hs1(keyword1)Σ $\psi_{keyword1}$, hs2(keyword1)Σ $\psi_{keyword1}$, ….)

Keyword2→(hs1(keyword2)Σ $\psi_{keyword1}$, hs2(keyword2)Σ $\psi_{keyword2}$, ….)

Index no 1

Keyword (i)→(hs1(keyword(i))Σ $\psi_{keyword1}$, hs2(keyword(i))Σ $\psi_{keyword1}$, ….)

Keyword (i+1)→(hs1(keyword(i+1))Σ $\psi_{keyword1}$, hs2(keyword(i+1))Σ $\psi_{keyword2}$, ….)

.

Index no Z

Keyword (z)→(hs1(keyword(z))Σ $\psi_{keyword1}$, hs2(keyword(z))Σ $\psi_{keyword1}$, ….)

Keyword (z+1)→(hs1(keyword(z+1))Σ $\psi_{keyword1}$, hs2(keyword(z+1))Σ $\psi_{keyword2}$, ….)

And Index Spatial

Index no 0

hs1→(keyword1)Σ Prob(keyword1), keyword2)Σ Prob(keyword2) , ….)

hs2→(keyword1)Σ Prob(keyword1), keyword2)Σ Prob(keyword2) , ….)

Index no 1

Hs i+1→(keyword1)Σ Prob(keyword1), keyword2)Σ Prob(keyword2) , ….)

Hs i+2→ (keyword1)Σ Prob(keyword1), keyword2)Σ Prob(keyword2) , ….)

So on.

With the formation of the indexes the hs quad tree is recalculated for the weight of each hs node and will rearrange all the node with same index will be presented under the same hs node. With this the distance of the location “hs code” from the parent “hs code” will be recalculated considering the distance mentioned by the query. Each hs_24 parent node will present the median of the global probable keyword and weight of keyword mentioned in the query will be matched with median of parent node and a desirable node will be identified and all the associated “h_s code” and their respective global probable keywords will present the result of the spatial search query

7.1 RESULTS

The CBDFI model spatial search algorithm was tested on airplane dataset from year 1975-2019, batches of 5 years each. Around 2500 different airplanes dataset was tested and 3456 associated text files defining the characteristic of the airplane investigation was recorded. Form this dataset 11675 different location were captured and total of 6704 unique h_s code were generated. Form the textual description 87657 unique keywords were recoded. All six indexed values from the p alpha{} universal set is checked and value 73 is selected for this hypothesis, hence 73 indices were generated for 1200 each keywords for textual description and 92 hs code each for location index.

Different element of the CBDFI model is tested for the given dataset to validate the outcomes of the proposed search algorithms. Various element of the CBDFI model such as a) hy-s indexer, b) hs encoder, c) Hs-I tree, d) pre-fetch table and e) search algorithm are tested with conventional system on the basis of operating system, database, number of keywords & indices and characteristic of the inputted query (search keyword and spatial-textual relevance). The following section highlight the result of CBDFI model.

The indexer plays an important role in any search algorithm. The performance of the indexer is dependent of the system configuration, hence the hybrid indexer is tested on different operating system i.e. Windows, Linux and Mac for variable RAM (4GB, 8GB, and 16GB). Table 7-1 list the configuration of the systems used to test hybrid indexing algorithm with inverted index, hash table.

Table 7-1 Configuration of different system used

SYSTEM	RAM (GB)	PROCESSOR
LINUX	4	AMD A-67310
LINUX	8	i5
LINUX	16	i7
WINDOWS	4	i3 5th gen
WINDOWS	8	i3 5th gen
WINDOWS	16	i3 5th gen
MAC OS	8	i5

All three indexing algorithms were compared for the execution time to process 87657 keywords and Table 7-2, 7-3, and 7-4 present comparison of execution time of the indexing algorithm i.e. inverted index, hash table and hybrid combine hash inverted index deployed on different operating systems.

Table 7-2 Execution time for Linux

RAM (Gb)	Inverted Index (s)	Hashing (s)	Combine hybrid (s)
4	14.397785	1.490357	14.611997
8	3.198658	0.593466	3.227291
16	2.341510	0.439	2.384156

Table 7-3 Execution time for Windows

RAM (GB)	Inverted Index (s)	Hashing (s)	Combine Hybrid (s)
4	6.386	1.245	5.785
8	3.118	0.522	2.883
16	1.499	0.254	1.2

Table 7-4 Execution time for Mac

RAM (GB)	Inverted Index (s)	Hashing (s)	Combine Hybrid (s)
8	3.864633	0.9532408	3.846816

From the three operating systems, Windows (i3 5th gen) operating system with 8GB RAM is selected to test the performance of the CBDFI search algorithm. This system configuration will be basic requirement for use of CBDFI model to search spatial objects. The systems were tested for varied number of keywords and the presented result is verified index number 73. The following graph will show case the performance of the combine indexing algorithms in term of execution time with increasing RAM size. With the increase in the RAM size better performance and less execution time of algorithms is encounter which can be explained in figure 7-1 for the Linux environment and figure 7-2 for the Windows environment. And figure 7-3 showcase the overall performance of all the three indexing algorithm for all the operating systems.

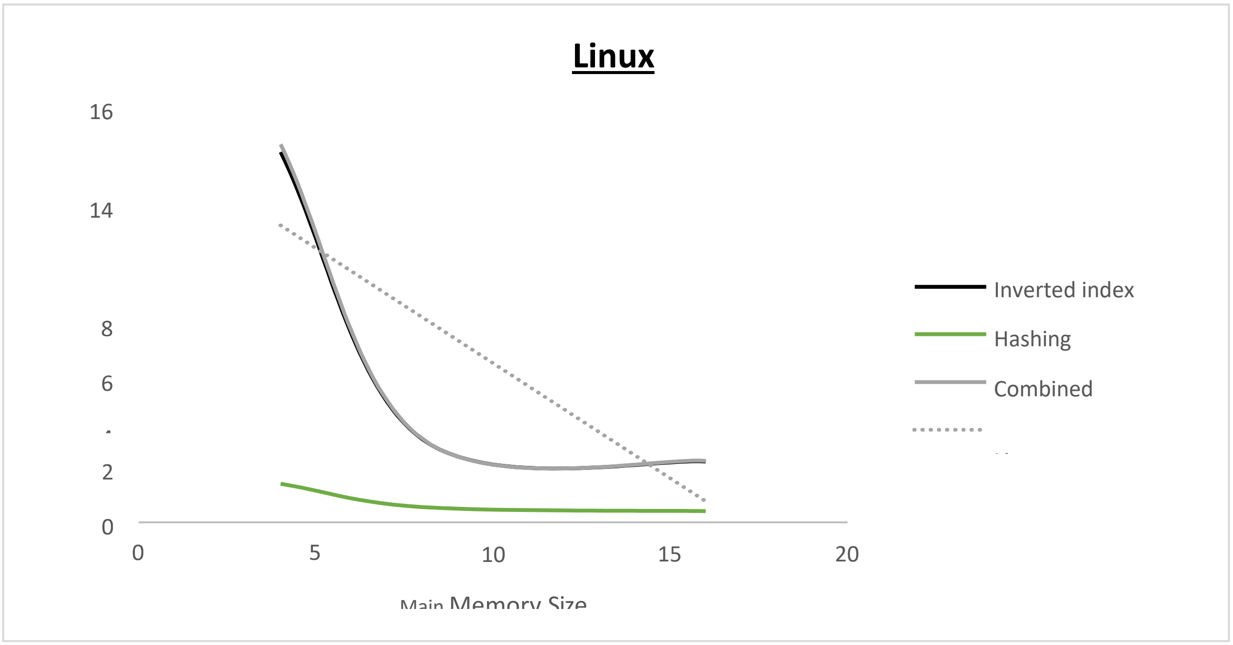


Fig 7-1 Performance of indexes on Linux

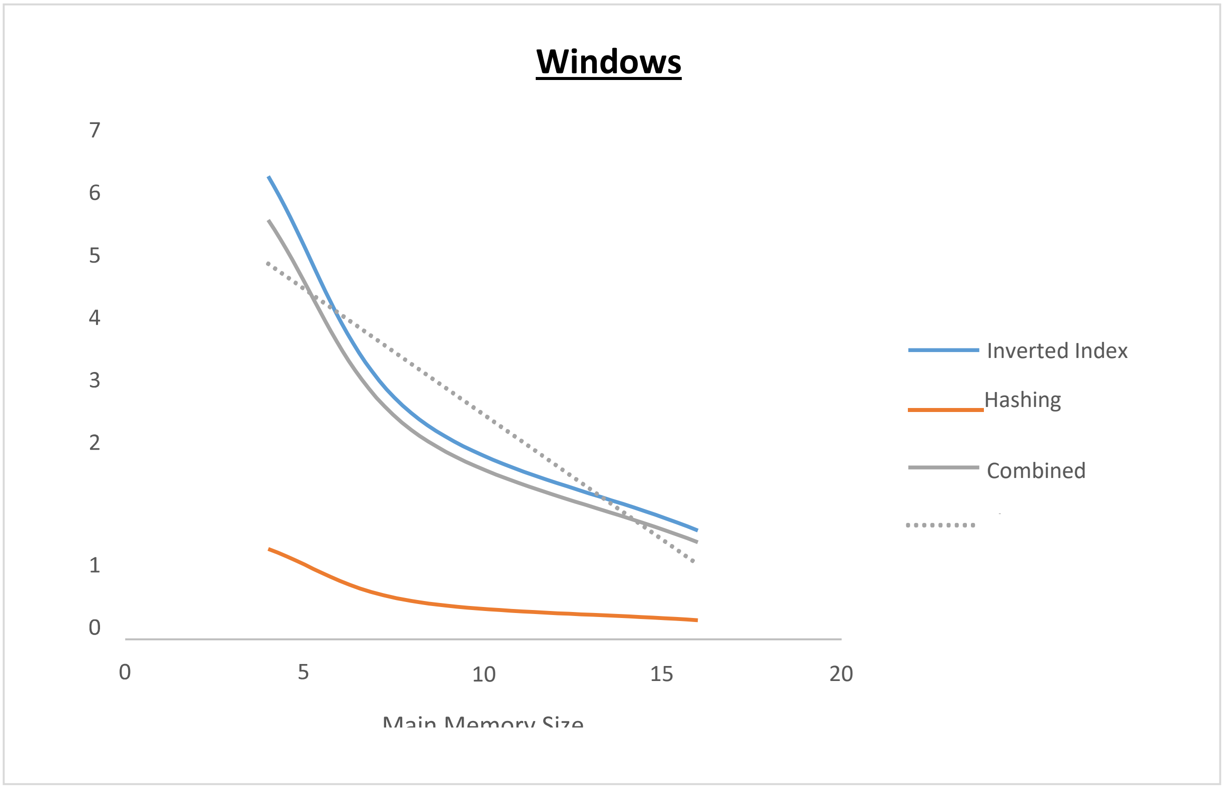


Fig 7-2 Performance on Indexes on Windows

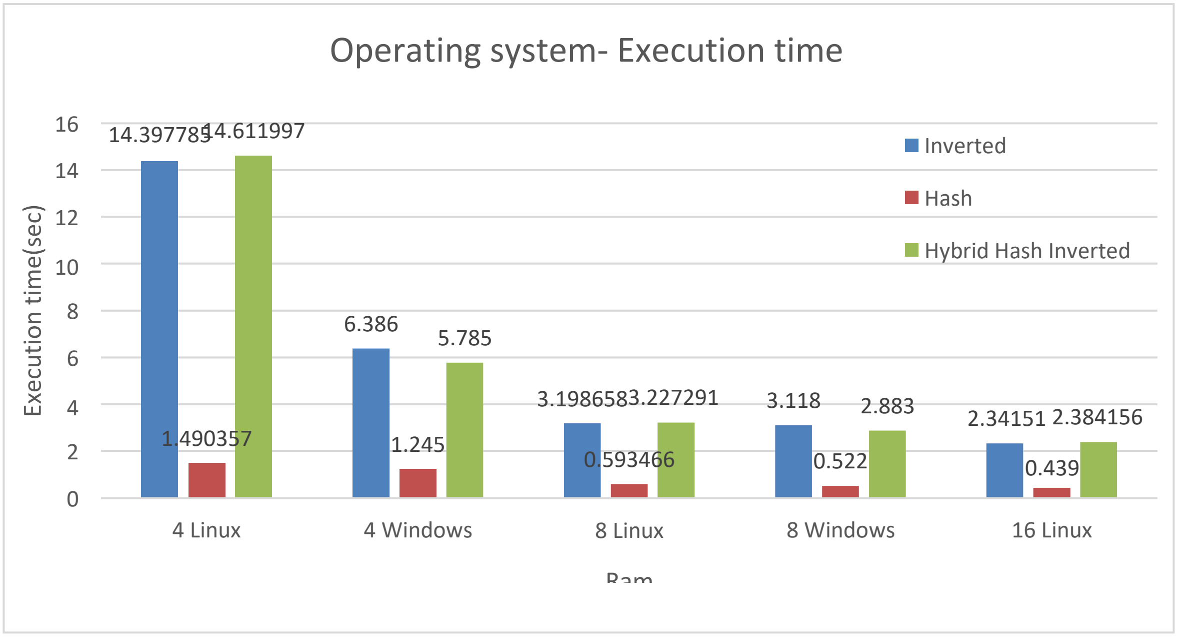


Fig 7-3 : Consolidated Performance of all the indexing algorithm

CBDFI model thrives on the pre-fetch table, where the global probable keywords are stored. Adaptive asynchronous pre-fetching algorithm was deployed on Window operating system and tested for three different databases i.e. GeoDatabase (PostGRE SQL), SQL database (Oracle 11g) and Column Database (MonetDb). The figure 7-4 showcase the time taken by the three database to deploy pre-fetch table based on number of keywords.

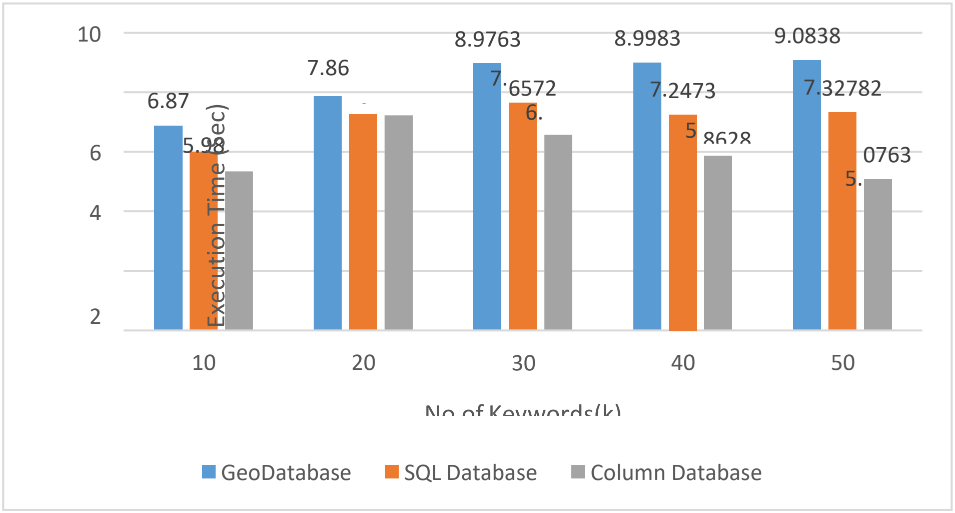


Fig 7-4 : Execution Time for formation of Pre-fetch table

The hybrid indexer will be used for the textual keyword as well as for the location keywords too. For processing the location using hash inverted index, h_s code of the respective location will be generated using h_s encoder. This h_s encoder will be compared with the conventional location encoder Geohash code. The comparison will be done on the basis of the distance from the user location for which the location code i.e. Geohash and Hs code need to be created. The table 7-5 showcase the performance comparison of Geohash code and Hs code.

Table 7-5 GeoHash vs. Hs code Generator

Distance (m)	**GeoHash(ns)**	**Hs code(ns)**
10	2.5678	9.1231
50	5.1435	10.1753
100	8.9657	11.6983
200	10.1098	12.111
500	12.5675	13.2363
750	13.2543	14.9776
1000	15	15.8776
1500	16.1765	16.8734
2000	17.2897	17.1
4000	18.6275	17.8948
8000	19.7765	19.6874
10000	21.4654	19.9864
20000	23.2876	22.6547
40000	27.1453	26.9976

100000	31.6756	31.4764

From figure 7-5, it can be concluded that h_s code generator take comparatively less execution time to generate location code then Geohash considering large distance from the user location. From the hypothesis and the experiment can validate the reason for better performance of h_s code is convergence of 24 location at single hs-24code. Therefore the CBDFI model produce better result for large distance dataset hence the airplane fleet management system work best with CBDFI model.

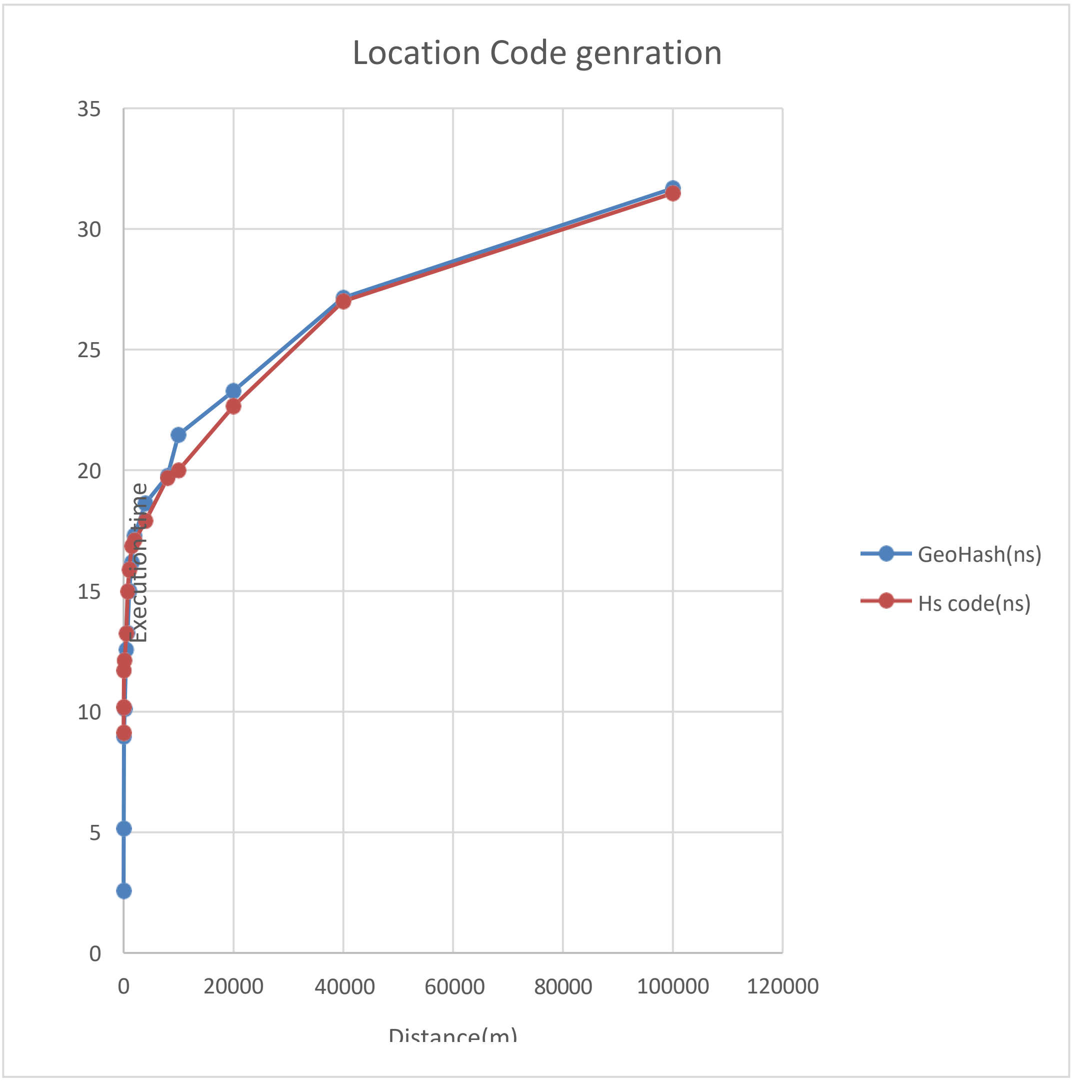


Fig7-5: Execution time taken by Geohash and Hs code to create location code

The hybrid indexer is now compared with textual indexer i.e. inverted index and hashing as well as for spatial indexer i.e. Geohash. The following table and graph showcase the performance of the hy-s indexer in context of the execution time. The table 7-6 present the performance of the indexing algorithm with respect to no of keywords registered for textual indexing, and figure 7-6 show execution time vs. no of keywords graph.

Table 7-6 Performance comparison of indexing algorithm depending on keywords

No of keyword	**Hashing(s)**	**Inverted Index(s)**	**Hybrid Hash Inverted Index (s)**
5	0.02	0.01	0.01
25	0.07	0.04	0.12
50	0.12	0.15	0.13
67	0.27	0.45	0.15
90	0.34	0.49	0.18
135	0.43	0.78	0.24
165	0.55	1	0.27
270	0.64	1.6	1.7
590	0.78	4.7	4.9
680	0.98	5.1	5
790	1.4	9.1	7.9
1075	1.8	9.8	9.9
1548	1.9	10.1	9.8
2598	2.1	10.6	9.7
4789	2.6	12	10.9
9876	2.8	13.5	12.8
15600	2.9	13.6	12.9
34679	3.1	13.4	13
75342	3.4	13.2	13.2
87567	3.45	13.29	13.09

From the figure 7-6 it can easily be identified that the proposed index Hy-s Indexer give better execution time with increase in the number of keywords and hence it is suitable when working with spatial big dataset. The performance can be future improve with more number of dataset keyword available for indexing.

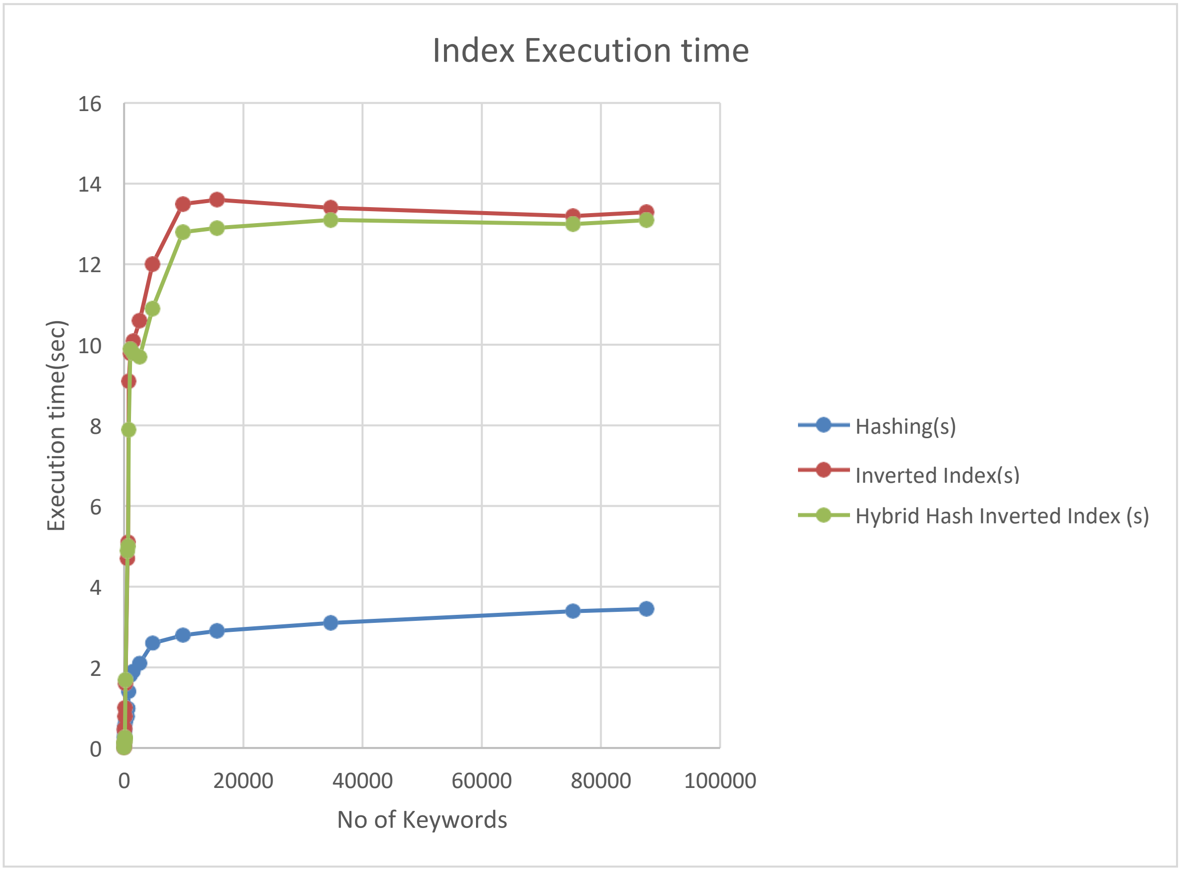


Fig 7-6: Indexing algorithms: No of keyword vs. execution time graph

The table 7-7 present the performance of the indexing algorithm with respect to no of h_s code keywords registered for spatial indexing, and figure 7-7 show execution time vs. no of keywords graph. The hy-S indexer is compared with the Geohash to validate the execution time taken by the proposed indexer while working with spatial keywords. The figure7-7 will identified and verify that the proposed indexer can work simultaneously for the textual indexing as well as spatial indexing and produce result that can be incorporated together in a quad tree and present a consolidated spatial quad tree with global probable keywords associated with each hs-24 node in the Hs-I tree.

Table 7-7 Comparison of geohash and hy-s indexer w.r.t keywords

No of keyword	Geohash(ns)	Hy-S Indexer
50	0.23	0.28
135	0.56	0.56
165	0.6	0.61
270	0.67	0.65
590	0.87	0.8
680	0.98	0.86
790	1.32	1.23
1075	1.5	1.34
1548	1.7	1.45
2598	2.1	2
4789	5.3	4.8
6074	5.8	5.1

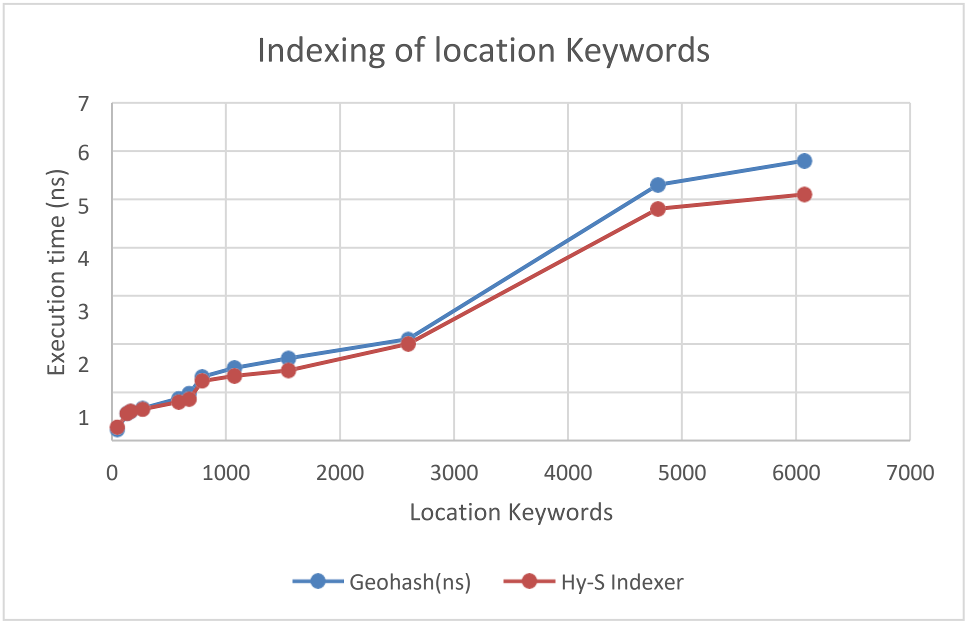


Fig 7-7: Execution time vs. keyword graph of Hy-s Indexer

P alpha{} universal set plays the most crucial role in improving the performance of the hybrid spatial indexer. If the number of the indices to be created is pre-assumed then the performance of the indexer can be monitored more efficiently. The table 7-8 and figure 7-8 show the performance of the Geohash indexer and hy-s indexer with respect to the time taken to execute index with increasing number of indices.

Table 7-8 Comparison of geohash and hy-s indexer (wrt to Indexes)

No of Index	Geohash	HI-S Indexer
2	0.01	0.011
3	0.02	0.024
5	0.02	0.024
7	0.02	0.034
11	0.03	0.031
13	0.04	0.051
17	0.09	0.052
19	0.1	0.079
23	0.11	0.092
29	0.23	0.19
31	0.34	0.4
37	0.36	0.41
41	0.45	0.45
43	0.46	0.49
47	0.47	0.53
53	0.51	0.61
59	0.53	0.67
61	0.54	0.63
67	0.67	0.53
71	0.89	0.51
73	0.87	0.61
79	0.91	0.78
83	0.94	0.82
89	1.06	0.91
97	1.11	1.3

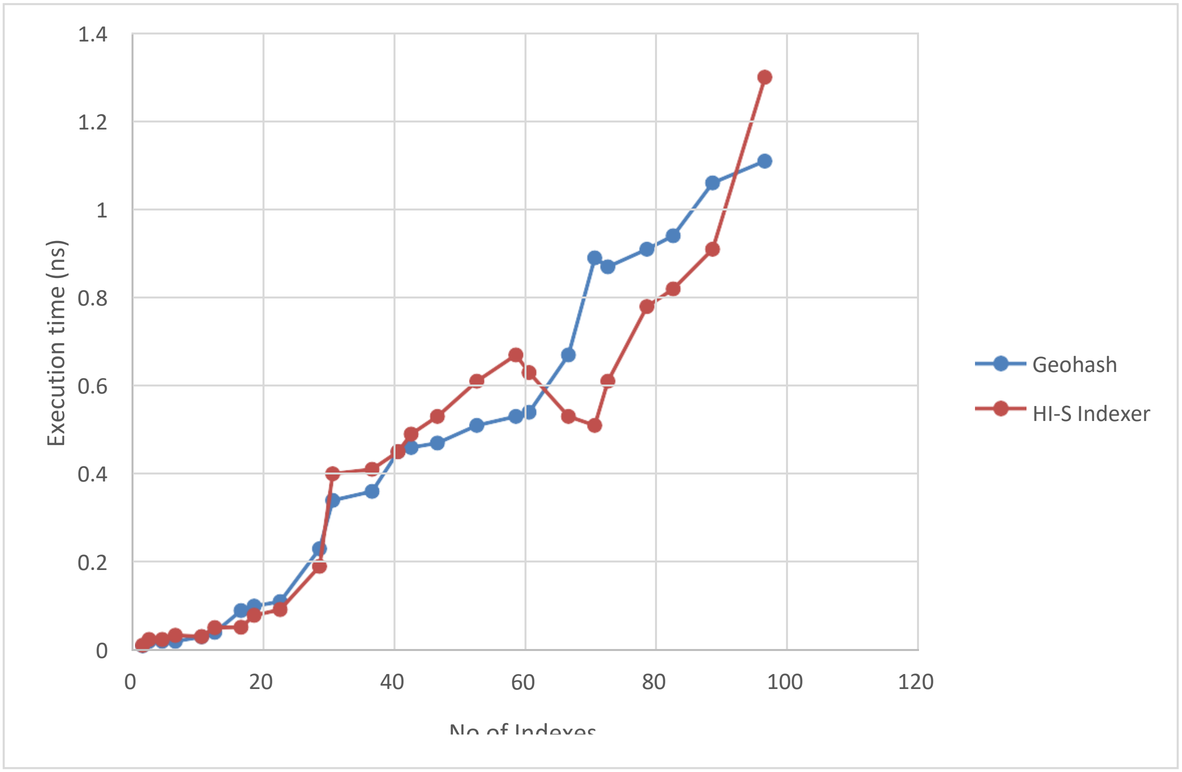


Fig 7-8 Execution time vs. no of Indices graph of Hi-S indexer

The Hy-S indexer is deployed in Hi-S tree to formulate spatial and textual indexer that can be used in top-k spatial keyword search query. Conventionally there are three major search algorithm in market: 1) IL- quadtree TopK- SK, 2) S2I –tree TopK- SK, and 3) I3- QuadTree TopK- SK. Table 7-9 showcase the comparison between the existing 3 spatial search algorithms.

Table 7-9 Comparison of Existing Spatial search algorithms

Search Algorithm	**IL-Quadtree TopK- SK**	**S2I-Tree TopK- SK**	**I3-Quadtree TopK- SK**
Textual Index	Inverted Index	Inverted Index	Inverted Index
Data Structure	Quad-Tree	R-Tree	Quad-Tree
Location Format	Deg	DMS	DMS
Dataset	Spatial-Temporal	Spatial Textual	Spatial Textual
Indexing Format	Textual first	Textual First	Spatial First
Bounding Condition	Ranking Function	MBR	Threshold Value

These search algorithm use similar concept to search spatial object through Top-K spatial keyword search query using combine spatial and textual index. The CBSFI search algorithm will be compared with these 3 conventional search algorithm to validate the performance of the proposed search algorithm w.r.t. the no of keywords, value of k and spatial and textual relevance a. The table 7-10 will showcase the performance of the Hs-I tree indices with existing conventional indices

Table 7-10: Comparison of Hi-S tree Execution time vs. No of Keywords

No of keyword	S2I tree	IL Tree	I3 Tree	Hi-S Tree
50	0.36	0.31	0.34	0.3
135	0.578	0.512	0.565	0.5
165	0.87	0.76	0.8	0.71
270	1.05	0.98	1.01	0.97
590	3.65	3.01	3.54	2.98
680	4.45	4.0342	4.87	3.98
790	4.87	4.78	4.98	4.54
1075	6.34	6.12	6.31	6.01
1548	7.87	7.67	7.678	7.54
2598	10.76	10.76	9.87	9.67
4789	15.76	13.67	13.21	12.98
9876	26.86	22.087	23.67	21.87
15600	39.87	35.67	32.87	31.876
34679	51.98	49.78	45.87	45.76
75342	98.87	90.09	85.87	81.78
83456	110	101	94.87	87.87
87657	167.89	159	163	155

From figure 7-9 it can be easily identified that proposed index Hs-I tree perform better as the number of keywords for indexing keep on increasing.

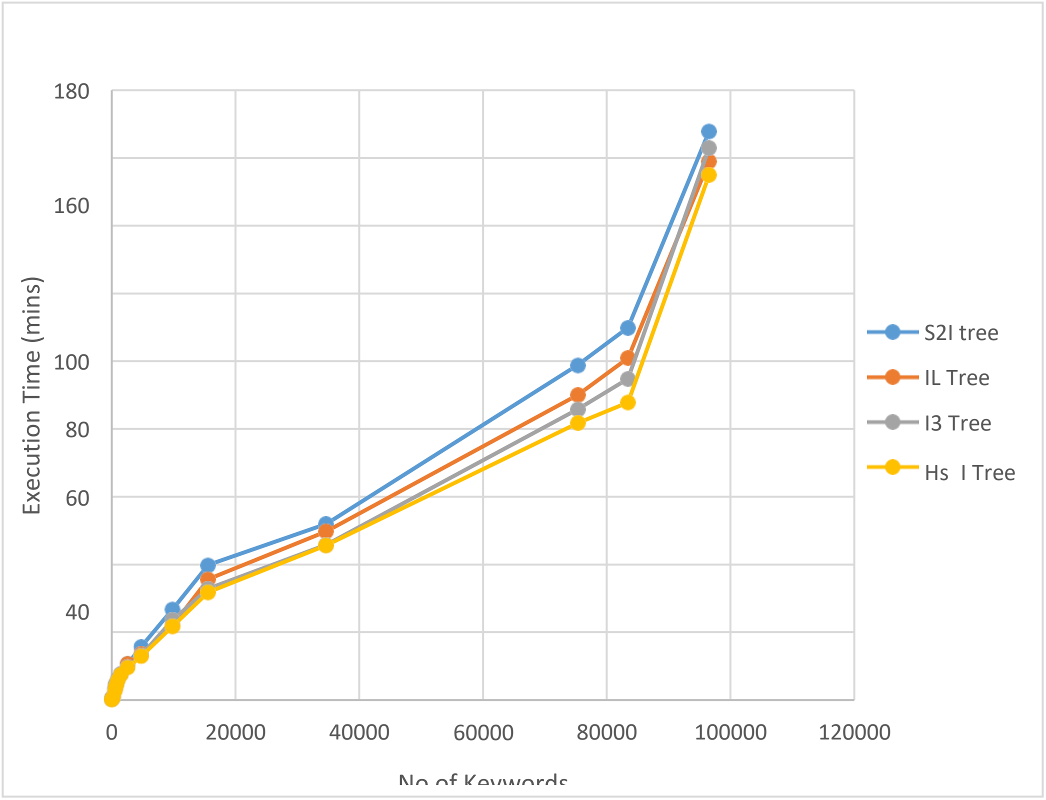


Figure 7-9: Performance comparison graph of various Search algorithms

The performance of the indices is also compared for the all the element of the p alpha {} universal set table 7-11 and figure 7-10 and then best performance for the CBDFI model was registered at value 73.

Table 7-11: Comparison of CBDFI search algorithm w.r.t. P alpha{}

No of Indices	67	71	73	79	83	89
Hi-S (65k Keywords)	72.72	70.87	69.98	71.09	73.82	75.937
Hi-S (75k Keywords)	85.982	83.762	81.9838	83.872	85.8633	87.863
Hi-S (85k Keywords)	92.988	90.762	88.98	91.0973	95.863	96.863
Hi-S (95k Keywords)	159.87	157.86	155.897	156.87	158.89	157.89
Hi-S (105k Keywords)	164.98	161.788	158.988	159.98	163.8776	167.97

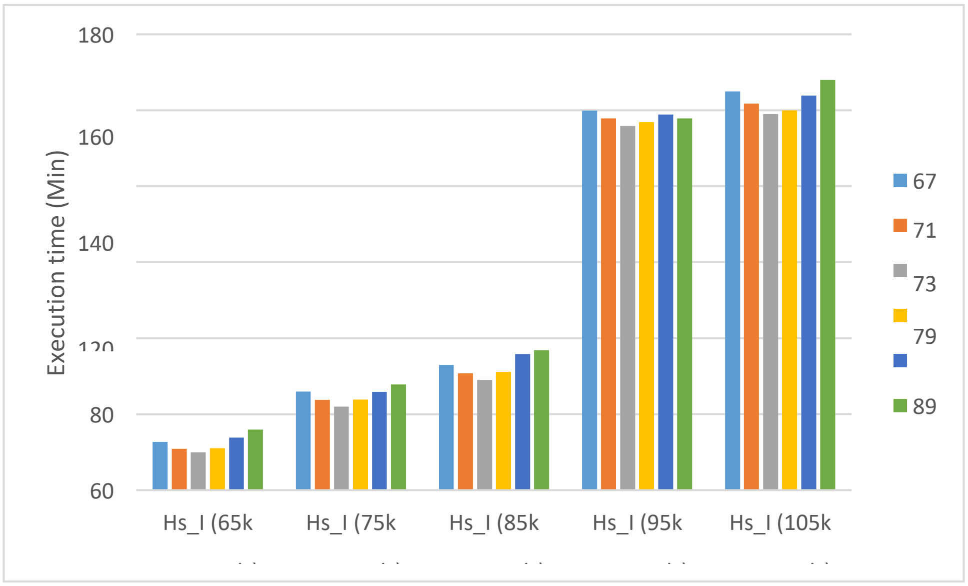


Fig 7-10: No. of keyword vs. Execution time of CBDFI model for p alpha {}

The table7-12 and 7-13 present a comparative graph comparing top k search based on SI-tree, I3 tree and Hs-I tree and show case the execution time query to search a desirable result. And the fetch time even keep on reducing with more number of queries been fetched.

Table 7-12 Comparison of Search algorithms (wrt spatial vs. textual relevance)

alpha a	SI-Tree(sec)	I3- tree(s)	Hi-S tree(s
0.1	4.6767	2.6762	2.6735
0.2	4.2355	2.7653	2.9098
0.3	3.6762	3.7624	3.02343
0.4	4.7582	4.7292	3.9876
0.5	5.6782	5.0187	3.8656
0.6	5.7686	5.2827	3.562
0.7	4.4673	5.8622	2.87783
0.8	3.5752	5.7872	2.6754
0.9	2.3567	6.982	2.3576

Table 7-13 Comparison of Search algorithms (wrt k)

k	SI-Tree(sec)	I3- tree(s)	Hi-S tree(s
1	5.8762	5.7692	3.75822
3	4.7682	4.54383	3.9792
5	3.6729	3.4563	3.2108

7	2.6582	2.9882	3.6827
9	2.8782	2.9882	4.6827

The figure 7-11 and 7-12 show the overall result of the CBDFI search algorithm

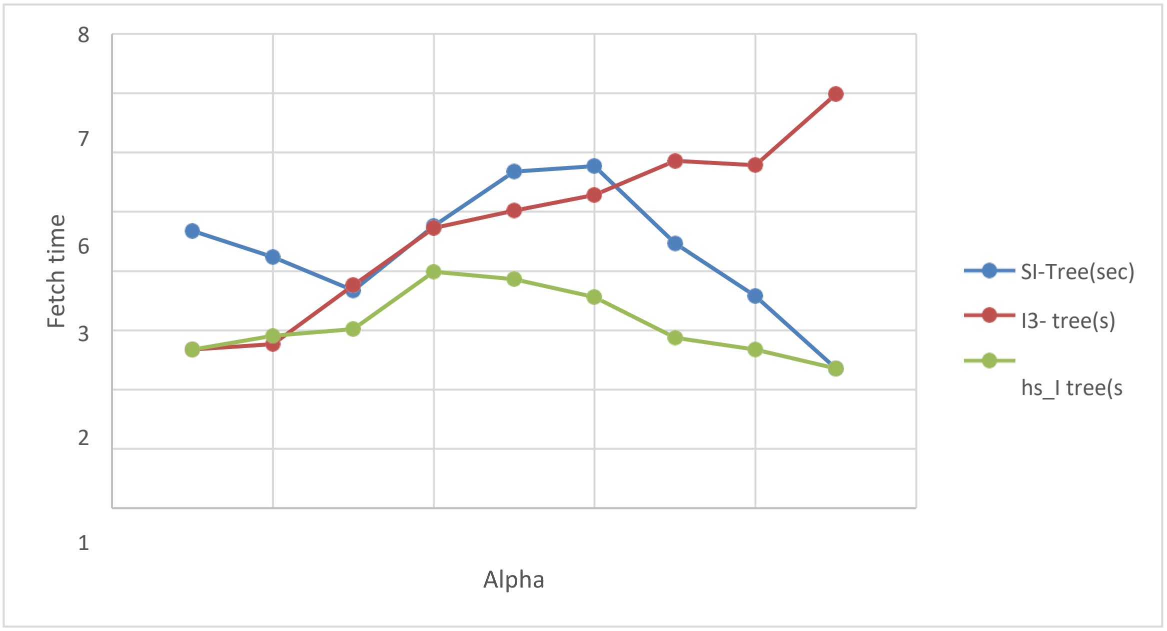


Fig 7-11: Fetch time of search Algorithm w.r.t. spatial-textual relevance

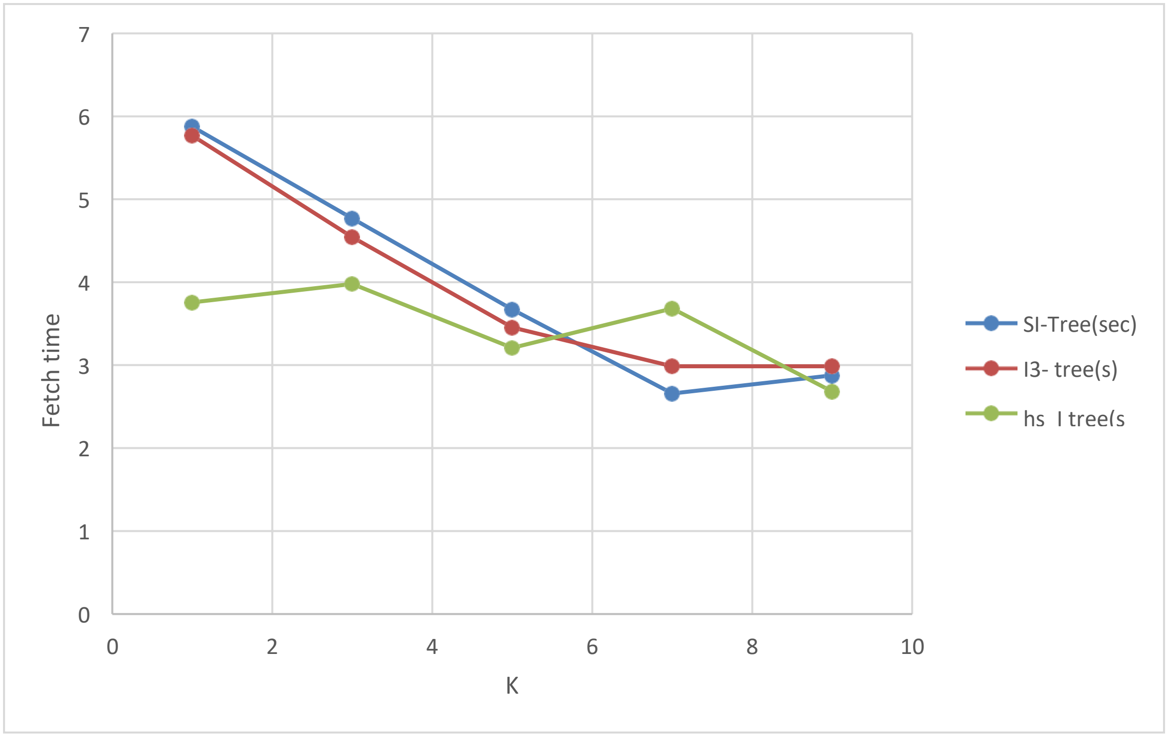


Fig 7-12 Fetch time of search algorithm w.r.t. to k

7.2 DISCUSSION

Combining Inverted index with hash table produce more accurate and efficient index. The same two algorithm are combined and tested in the presented CBDFI model. From the various experiment conducted on the dataset of aviation industry it can be tested and verified that the combine model yield 23% faster result while searching the database using index based on hashed inverted index. Another significant finding from the research is the generation of h_s code which produce a consolidated 7 bit hexadecimal code including both latitude and longitude of any spatial objects. Comparing with the successor Geohash ,hs code is approximately 23% compressed and produce best result with considering equal relevance of spatial and textual score. The CBDFI model produce most optimal result at k = 5 and table 7-14 showcase the complete finding of the proposed model.

Table 7-14: Finding of the CBDFI Search algorithm

S.No	Steps	Conclusions	Findings
1	Study the architecture of column based database	CBDFI Model for Search of spatial Data in Column Database	• Designed for hetro-type spatial data. • Location Logical view is optimized using HMM
2	Process location data	Unique Hs Code	• Hs code is 23.33 % compressed then Geohash code • Execution time is approximately equal to Geohash. • More the distance buffer better results. • Average time (3.45672 sec) taken by hs code to cover 1 UTM zone is better than geohash
3	Identify algorithm used to index spatial data	Hy-S Indexer	• The indexer work best in 8 GB windows environment. • Same Index can be used for location as well as textual keywords • Indexer work optimal when the

			hash table index in range of p alpha{} • The indexer start giving better results when the number of keywords are more than 20000, approximately 17.768%
4	Define spatial search model and design novel search algorithm.	Hs-I tree based spatial search algorithm	• Yield most optimal result at k =5 and equal spatial and textual relevance. • At equal relevance the proposed search algorithm is average 26.544 % fast in executing • Hs-I tree is 22.97% i3-tree and 31.92% faster the Si-Tree at equal relevance
5	Compare performance of proposed search algorithm	Pre-Fetch table and its improving efficiency with increasing queries	• Minimum time is taken to render pre-fetch table designed in window with Monet db, compared with postgre Sql and Oracel 11g • The proposed search query is tested for a keyword based search.

CHAPTER 8

CONCLUSION & FUTURE SCOPE

8.1 CONCLUSION

The similitude of Big-data and location aware data has evolved the concept of "Spatial Big Data" (SBD). For the processing of the spatial big data, advancement in the conventional technologies is a demand of the industry. Multiple different solution has been presented by researchers globally to optimize the manipulation of SBD with various different kind of database storage architecture. The presented CBDFI model is also one such contribution in the field of spatial data processing to optimize the search seek time. The contemporary search query handler i.e. "Top K" spatial keyword search query which provide efficient result for batch processing, moving trajectory and distributed processing is used as the base search engine for the current research. "CBDFI" model proposed a hybrid indexing algorithm which can process both keyword and location simultaneously. This "Hy-S" indexer use a novel location encoder which produce unique h_s code for any inputted location. Hidden markov model is used as the base mathematical structure to traverse among the different "hs code" and generate the rule for traversing from one node to another of "Hs-I Quadtree". The "hs encoder" produce a group of 24 different location that can be processed simultaneously at the single node of the "Hs-I Quadtree" hence reducing the search time of the spatial keyword. CBDFI model also present Pre-Fetch table which is searched for the most frequent keywords and positive keyword match in this list will make the time complexity of the algorithm of O(m) where m is the size of the list.

The CBDFI model performance is based on the hybrid hash inverted indexer, hence this indexer was compare with the other two indexer i.e. inverted index and hash table on various different operating systems and the hash inverted index produced 7.5 % improvement from the contemporary inverted index for Windows I3 5th gen, 8 GB system. Bloom filter is another contemporary hash table used for searching keywords from queries but it work only on textual keyword whereas presented hash inverted work on both location and textual keyword.

The Prefetch table was deployed on various databases such as PostGreSQL, Oracle 11g and MonetDB. Hence, Prefetch designed for CBDFI model on MonetDB produce 23.11 % better results than Oracle 11g and 37.4 % better result than PostGreSQL for 73 keywords.

The CBDFI model compared with three different spatial search model i.e. IL-Quadtree Topk-Sk, S2I-Quadtree Topk-SK and I3-Quadtree Topk-Sk and two major drawback were identified. Firstly the search engine use two different indexer for location and textual keywords where as CBDFI model can use the same Hy-S indexer for both textual and location keywords simultaneously. Secondly the search engine use Degree decimal and DMS format for location information where as CBDFI model use h_s code.

The CBDFI model search algorithm was compared with S2I-Quadtree and I3-Quadtree and it was identified that model run best for value of alpha .5 i.e. equal spatial and textual relevance and for the value of k= 5 in Top k query.

The CBDFI model introduce h_s code and unique location code and datatype for Monet db. Hence h_s code is 23.33 % compress then Geohash codes. More the distance to cover the better result are showcased by h_s code and an average time of 3.45672 second is taken to cover one UTM zone

The CBDFI Model is approximately 26.544 % faster in execution and 22.97% faster than I3-Quadtree and 31.92 % faster than S2I-Quadtree. The model showcase approximately 17.768 % improvement in result as the keyword increase more than 20000.

The presented research can be used in various different application area such as resource planning, social media surveillance, travel planning, crime investigation, and trajectory monitoring and cyber threats. With the use of h_s code the storage cost as well as processing time for spatial data can be improved and can produce better location aware service for Industry 4.0

8.2 FUTURE SCOPE

The presented framework has been designed around Column database architecture but it might provide better result while working with graph database. The presented research can be extended to produce location tool based on h_s code which can be fabricated in a hardware using universal gates and Boolean algebra. And with the advancement of artificial neural network, the proposed transition matric may produce better results if concepts of Artificial Intelligence is used.

REFERENCES

Abramova, V., Bernardino, J.j & Furtado, P.(2014). Which NoSQL database?. A performance overview. Open Journal of database (OJDB), 1(2), 17-24.

Alvares, L. O., Bogorny, V., Kuijpers, B., de Macedo, J. A. F., Moelans, B., & Vaisman, A. (2007). A model for enriching trajectories with semantic geographical information. Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems - GIS '07. doi:10.1145/1341012.1341041

A.Cary, O. Wolfson, and N. Rishe, "Efficient and scalable method for processing top-k spatial boolean queries," in SSDBM, 2010, pp. 87–95. Doi: 10.1007/978-3-642-13818-8_8

Khodaei, C. Shahabi, and C. Li. Hybrid indexing and seamless ranking of spatial and textual features of web documents. In DEXA (1), pages 450– 466, 2010. Doi: 10.1007/978-3-642-15364-8_37

Barewar, A., Radke, M. A., & Deshpande, U. A. (2014). Geo skip list data structure - storing spatial data and efficient search of geographical locations. 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI). doi:10.1109/icacci.2014.6968370

Barowy. W. Daniel .et .al flashrelate(2015): Extracting relational Data from Semi-structured spreadsheet using example. PLDI'15, June 13-17, 2015, Portland, Or, USA, ACM978-1-4503-3468-6/15/06 doi: 10.1145/2813885.2737952

Bridget Freisthler, Bridgette Lery, Paul J. Gruenewald, and Julian Chow: Methods and Challenges of Analyzing Spatial Data for Social Work Problems: The Case of Examining Child Maltreatment Geographically, doi 10.1093_swr_30.4.198

Bu, Y., Howe, B., Balazinska, M., & Ernst, M. D. (2010). HaLoop. Proceedings of the VLDB Endowment, 3(1-2), 285–296. doi:10.14778/1920841.1920881

Cao, X., Cong, G., Jensen, C.S.: Retrieving top-k prestige-based relevant spatial Web objects. Proc. VLDB Endowment 3(1-2), 373–384 (2010)

Cao, X., Cong, G., Jensen, C. S., & Ooi, B. C. (2011). Collective spatial keyword querying. Proceedings of the 2011 International Conference on Management of Data - SIGMOD '11. doi:10.1145/1989323.1989363

Chen, Y.-Y., Suel, T., & Markowetz, A. (2006). Efficient query processing in geographic web search engines. Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data - SIGMOD '06. doi:10.1145/1142473.1142505

Christine Jardak, Petri Mähönen, and Janne Riihijärvi (2014): Spatial Big Data and Wireless Networks: Experiences, Applications, and Research Challenges, 0890-8044/14/$25.00 © 2014 IEEE, doi 10.1109_MNET.2014.6863128

Christoforaki, M., He, J., Dimopoulos, C., Markowetz, A., Suel, T.: Text Vs. Space: Efficient Geo-Search Query Processing. In: Proceedings of the 20Th ACM International Conference on Informationand Knowledge Management, pp. 423–432 (2011)

Chu, Xu, John Morcos, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, and Nan Tang. (2015) "KATARA: A Data Cleaning System Powered by Knowledge Bases and Crowdsourcing", in Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia.

Cong, G., Jensen, C. S., & Wu, D. (2009). Efficient retrieval of the top-k most relevant spatial web objects. Proceedings of the VLDB Endowment, 2(1), 337–348. doi:10.14778/1687627.1687666

Corbellini, A., Mateos, C., Zunino, A., Godoy, D., & Schiaffino, S. (2017). Persisting big-data: The NoSQL landscape. Information Systems, 63, 1–23. doi:10.1016/j.is.2016.07.009

Costes and J. Perret, “A hidden Markov model for matching spatial networks,” Journal of Spatial Information Science, no. 18, Jun. 2019, doi: 10.5311/josis.2019.18.489.

C. Zhang, Y. Zhang, W. Zhang and X. Lin, "Inverted Linear Quadtree: Efficient Top K Spatial Keyword Search," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1706-1721, 1 July 2016, doi: 10.1109/TKDE.2016.2530060.

Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. Communications of the ACM 51(1), 107–113 (2008)

De Felipe, I., Hristidis, V., & Rishe, N. (2008). Keyword Search on Spatial Databases. 2008 IEEE 24th International Conference on Data Engineering. doi:10.1109/icde.2008.4497474

Dong-Wan Choi , Chin-Wan Chung (2015): Nearest Neighborhood Search in Spatial Databases, 978-1-4799-7964-6/15/$31.00 © 2015 IEEE, doi 10.1109_ICDE.2015.7113326

Doulkeridis, C., Nørvåg, K. A survey of large-scale analytical query processing in MapReduce. The VLDB Journal 23, 355–380 (2014). https://doi.org/10.1007/s00778-013-0319-9.

D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. Efficient olap operations in spatial data warehouses. In SSTD, pages 443–459, 2001.

D. Wu, M. L. Yiu, G. Cong, and C. S. Jensen, “Joint top k spatial keyword query processing,” TKDE, 2011

D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung, and M. Kitsuregawa. Keyword search in spatial databases: Towards searching by document. In ICDE, pages 688–699,2009.

D. Zhang, B. C. Ooi, and A. K. H. Tung. Locating mapped resources in web 2.0. In ICDE, pages 521–532, 2010.

D. Zhang, K.-L. Tan, and A. K. H. Tung, “Scalable top-k spatial keyword search,” in EDBT, 2013, pp. 359–370

Eskandari L, Huang Z, Eyers D (2016) P-Scheduler: adaptive hierarchical scheduling in apache storm.In: Proceedings of the Australasian Computer Science Week Multi -conference , ACSW 2016, No. 26. ACM Press, New York

Finkel, R. A., & Bentley, J. L. (1974). Quad trees a data structure for retrieval on composite keys. Acta Informatica, 4(1), 1–9. doi:10.1007/bf00288933

Georgiou, H., Pelekis, N., Sideridis, S., Scarlatti, D., & Theodoridis, Y. (2019). Semantic-aware aircraft trajectory prediction using flight plans. International Journal of Data Science and Analytics. doi:10.1007/s41060-019-00182-4

Ghemawat, S., Gobioff, H., Leung, S.: The google file system. In: ACM SIGOPS Operating Systems Review, vol. 37, pp. 29–43. ACM (2003)

Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D.: Trajectory Pattern Mining.

In: SIGKDD, pp. 330–339 (2007)

Gregory Giuliani, Nicolas Ray, Anthony Lehmann(2011): Grid-enabled Spatial Data Infrastructure for environmental sciences: Challenges and opportunities, Future Generation Computer Systems 27 (2011) 292–303

Griffith, D. A., Fischer, M. M., & LeSage, J. (2016). The spatial autocorrelation problem in spatial interaction modelling: a comparison of two common solutions. Letters in Spatial and Resource Sciences, 10(1), 75–86. doi:10.1007/s12076-016-0172-8

Guan, X., & Chen, C. (2014). Using social media data to understand and assess disasters. Natural Hazards, 74(2), 837–850. doi: 10.1007/s11069-014-

1217-1

Guo, L., Shao, J., Aung, H. H., & Tan, K.-L. (2014). Efficient continuous top-k spatial keyword queries on road networks. GeoInformatica, 19(1), 29–60. doi:10.1007/s10707-014-0204-8

H.-J. Hong, G.-M. Chiu, and W.-Y. Tsai, "A single quadtree-based algorithm for top-k spatial keyword query," Pervasive and Mobile Computing, vol. 42, pp. 93–107, Dec. 2017

Karnitis, G., & Arnicans, G. (2015). Migration of Relational Database to Document-Oriented Database: Structure Denormalization and Data Transformation. 2015 7th International Conference on Computational Intelligence, Communication Systems and Networks. doi:10.1109/cicsyn.2015.30

Khayyat, Zuhair, Ihab F. Ilyas, Alekh Jindal, Samuel Madden, Mourad Ouzzani, and Paolo Papotti. (2015) "BigDansing: A System for Big Data Cleansing", in Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia.

Kyu-Young Whang , Jae-Gil Lee , Min-Soo Kim, Min-Jae Lee ,Ki-Hoon Lee ,Wook-Shin Han ,Jun-Sung Kim(2010): Tightly-coupled spatial database features in the Odysseus/OpenGIS DBMS for high- performance, Geoinformatica (2010) 14:425–446 DOI 10.1007/s10707-009-0086-3

K. V. R. Kanth, S. Ravada, and D. Abugov. Quadtree and r-tree indexes in oracle spatial: a comparison using gis data. In SIGMOD Conference, pages 546–557, 2002.

Lee, J.-G., & Kang, M. (2015). Geospatial Big Data: Challenges and Opportunities. Big Data Research, 2(2), 74–81.doi:10.1016/j.bdr.2015.01.003

Li, Z., Lee, K. C. K., Zheng, B., Lee, W.-C., Lee, D., & Wang, X. (2011). IR-Tree: An Efficient Index for Geographic Document Search. IEEE

Transactions on Knowledge and Data Engineering, 23(4), 585–599. doi:10.1109/tkde.2010.149

Li, Y. Li, and M. L. Yiu, “A Spatial Insight for UGC Apps: Fast Similarity Search on Keyword-Induced Point Groups,” in 2019 20th IEEE International Conference on Mobile Data Management (MDM), 2019 doi: 10.1109/MDM.2019.00-26

Liu, F., Yu, C., Meng, W., & Chowdhury, A. (2006). Effective keyword search in relational databases. Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data - SIGMOD ’06. doi:10.1145/1142473.1142536

Liu Hua, Li De-ren, ZHU Xin-yan(2005): Large volume spatial Data management based on Grid computing IEEE 2005, 0-7803-9050-4/05/$20.00

Liu Xiaosheng, Huang Xiaobin, Zhoa Zhiyong(2008): Research on spatial Database Model in Grid environment : The international archive of photogrammetry, Remote Sensing and Spatial Information Science Vol. XXXVII part B4 Beijing 2008

LIU, Z., GUO, H., & WANG, C. (2016). Considerations on Geospatial Big Data. IOP Conference Series: Earth and Environmental Science, 46, 012058. doi:10.1088/1755-1315/46/1/012058

Low, Y., Gonzalez, J., Kyrola, A., Bickson, D.,Guestrin, C., Hellerstein, J.M.: Graphlab: A new framework for parallel machine learning. arXiv preprint arXiv:1006.4990 (2010)

Ls Carlo Strozzi, NoSQL Relational Database Management System: Home Page, URL http://www.strozzi.it/cgibin/CSA/tw7/I/en_US/nosql/Home-Page, 1998 (accessed 05.07.13).

Lu, J., Lu, Y., Cong, G.: Reverse Spatial and Textual K Nearest Neighbor Search. In: SIGMOD (2011)

L.M. Ainsworth, C.B. Dean, and R. Joy: Zero-Inflated Spatial Models: Application and Interpretation, DOI 10.1007/978-3-319-31260-6_3

Malewicz, G., Austern, M., Bik, A., Dehnert, J., Horn, I., Leiser, N., Czajkowski, G.: Pregel: a system for largescale graph processing. In: Proceedings of the 2010 international conference on Management of data, pp. 135–146.ACM (2010)

Markus Törmä, Pekka Härmä, Elise Järvenpää(2007) , Change Detection using Spatial Data:Problems and Challenges, 1-4244-1212-9/07/$25.00

©2007 IEEE. doi 10.1109_IGARSS.2007.4423208

Mohan, C. (2013). History repeat itself : Sensible and NonsenSQL aspectof the NoSQL hoopla. EDBT/ICDT 2013 Joint conference, March 18-22, 2013, Genoa Italy, ISBN: 978-1-4503-1597-5

Muhammad Hanis Rashidan, Ivin Amri Musliman(2015): GeoPackage as Future Ubiquitous GIS Data Format: A Review 73:5 (2015) 47–53 | eISSN 2180–3722 |

Murray, D.G., Schwarzkopf, M., Smowton, C., Smith, S., Madhavapeddy, A., Hand, S.: Ciel: a universal execution engine for distributed data-flow computing. In: Proceedings of the 8th USENIX conference on

M. Christoforaki, J. He, C. Dimopoulos, A. Markowetz, and T. Suel. Text vs. space: efficient geo-search query processing. In CIKM, pages 423–432, 2011

Nguyen, T., Larsen, M. E., O’Dea, B., Nguyen, D. T., Yearwood, J., Phung, D.,

… Christensen, H. (2017). Kernel-based features for predicting population health indices from geocoded social media data. Decision Support Systems, 102, 22–31. doi:10.1016/j.dss.2017.06.010

Ningyu Zhang, Guozhou Zheng, Huajun Chen (2014): HBaseSpatial: a Scalable Spatial Data Storage Based on HBase, 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and

Communications, 978-1-4799-6513-7/14 $31.00 © 2014 IEEE, DOI 10.1109/TrustCom.2014.83

Peter A. Johnson (2016): Models of direct editing of government spatial data: challenges and constraints to the acceptance of contributed data, Cartography and Geographic Information Science, DOI: 10.1080/15230406.2016.1176536.

Ridzuan, F., & Wan Zainon, W. M. N. (2019). A Review on Data Cleansing Methods for Big Data. Procedia Computer Science, 161, 731–738. doi:10.1016/j.procs.2019.11.177

Rigaux P, Scholl M, Voisard A (2002) 6 - spatial access methods.In: Spatial Databases, Morgan Kaufmann, San Francisco, pp 201–266, DOI http://dx.doi.org/10.1016/B978-155860588-6/50008-7, URL http://www.sciencedirect.com/science/article/pii/B9781558605886500087

Robert. T. Mason(2015) NoSQL database and Data Modeling Technique for a document oriented NoSQL Database. Proceeding of informing science & IT education conference (InSITE) 2015, 259-268 doi: 10.28945/2245

Rocha-Junior, J. B., Gkorgkas, O., Jonassen, S., & Nørvåg, K. (2011). Efficient Processing of Top-k Spatial Keyword Queries. Lecture Notes in Computer Science, 205–222. doi:10.1007/978-3-642-22922-0_13

Ruiz-Medina, M. D. (2012). New challenges in spatial and spatiotemporal functional statistics for high-dimensional data. Spatial Statistics, 1, 82–91. doi:10.1016/j.spasta.2012.02.006

R. A. Baeza-Yates and B. A. Ribeiro-Neto. Modern Information Retrieval. ACM Press / Addison-Wesley, 1999.

R. Hariharan, B. Hore, C. Li and S. Mehrotra, "Processing Spatial-Keyword (SK) Queries in Geographic Information Retrieval (GIR) Systems," *19th International Conference on Scientific and Statistical Database Management (SSDBM 2007),* 2007, pp. 16-16, doi:

10.1109/SSDBM.2007.22.

R. Prakhyath, M. Vijaya(2015): Transforming digital unstructured and semi structured data into structured data with aid of IE and KDT international Journal in Research in Computer and Communication technology Vol4,

Schade (2015) : BIG DATA BREAKING BARRIERS – FIRST STEPS ON A LONG TRAIL, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XL-7/W3, 36th International Symposium on Remote Sensing of Environment, 11–15 May 2015, Berlin, Germany DOI:10.5194/isprsarchives-XL-7-W3-691-2015

Seung Kyoon Shin, G. Lawrence Sanders (2006): Denormalization strategies for data retrieval from data warehouses, Decision Support Systems 42 (2006) 267– 282, doi 10.1016_j.dss.2004.12.004

Singh, B. & Srivastava, K. & Gupta, D.. Future prospects and challenges in geospatial database for handling of big data concept: A review. International Journal of Recent Technology and Engineering. 7. 140-144. April 2019

Song, W. & Jin, B. & Shihua, Li & Wei, X. & Li, D. & Hu, Fei. (2015). BUILDING SPATIOTEMPORAL CLOUD PLATFORM FOR SUPPORTING GIS APPLICATION. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences. II-4/W2. 55-62. 10.5194/isprsannals-II-4-W2-55-2015.

Song, Z., Chen, J., & Ye, J. Y. (2014). A Mobile Storage System for Massive Spatial Data. Advanced Materials Research, 962-965, 2730–2734. doi:10.4028/www.scientific.net/amr.962-965.2730

Srivastava, K., Sridhar, P.S.V.S. and Dehwal, A. 'Data integration challenges and solutions: a study', International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 2, No. 7, July 2012

Storey, V. C., & Song, I.-Y. (2017). Big data technologies and Management:

What conceptual modeling can do. Data & Knowledge Engineering, 108, 50–67.doi:10.1016/j.datak.2017.01.001

Storey, Veda & Chiang, Roger & Goldstein, Robert & Dey, Debabrata. (1997). Database Design with Common Sense Business Reasoning and Learning. ACM Trans. Database Syst. Doi: 22. 10.1145/278245.278246.

Sun, D., Yan, H., Gao, S., Liu, X., & Buyya, R. (2017). Rethinking elastic online scheduling of big data streaming applications over high-velocity continuous data streams. The Journal of Supercomputing, 74(2), 615–636. doi:10.1007/s11227-017-2151-2

Tsung-Hao Chen , Cheng-Wu Chen(2010): Application of data mining to the spatial heterogeneity of foreclosed mortgages, Expert Systems with Applications 37 (2010) 993–997, doi 10.1016_j.eswa.2009.05.076

Verena Kantere, Spiros Skiadopoulos, and Timos Sellis(2008): Storing and Indexing Spatial Data in P2P Systems, 1041-4347/09/$25.00 _ 2009 IEEE, DOI 10.1109_tkde.2008.139]

Wang, Hongzhi, Mingda Li, Yingyi Bu, Jianzhong Li, Hong Gao, and Jiacheng Zhang. (2015) "Cleanix: a Parallel Big Data Cleaning System." ACM SIGMOD Record 44 (4): 35-40. Doi: 10.1145/2935694.2935702

Yan, Z., Chakraborty, D., Parent, C., Spaccapietra, S., & Aberer, K. (2011). SeMiTri. Proceedings of the 14th International Conference on Extending Database Technology - EDBT/ICDT '11. doi:10.1145/1951365.1951398

Yakout, M., Berti-Équille, L., & Elmagarmid, A. K. (2013). Don't be SCAREd. Proceedings of the 2013 International Conference on Management of Data - SIGMOD '13. doi:10.1145/2463676.2463706

Ye, D. (2008). The evolution of geographic information systems from my view. Geoinformatics 2008 and Joint Conference on GIS and Built Environment: The Built Environment and Its Dynamics. doi:10.1117/12.812823

Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications, 1(1), 7–18. doi:10.1007/s13174-010-0007-6

Zhang, Y., Gao, Q., Gao, L., & Wang, C. (2011). PrIter. Proceedings of the 2nd ACM Symposium on Cloud Computing - SOCC '11. doi:10.1145/2038916.2038929

Zheng, K., Gu, D., Fang, F., Zhang, M., Zheng, K., & Li, Q. (2017). Data storage optimization strategy in distributed column-oriented database by considering spatial adjacency. Cluster Computing, 20(4), 2833–2844. doi:10.1007/s10586-017-1081-3

Zhou, Y., Xie, X., Wang, C., Gong, Y., & Ma, W.-Y. (2005). Hybrid index structures for location-based web search. Proceedings of the 14th ACM International Conference on Information and Knowledge Management - CIKM '05. doi:10.1145/1099554.1099584

Zhang, N., Zheng, G., Chen, H., Chen, J., & Chen, X. (2014). HBaseSpatial: A Scalable Spatial Data Storage Based on HBase. 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications. doi:10.1109/trustcom.2014.83

APPENDIX: A

WORKS PUBLISHED/PRESENTED/ACCEPTED FOR PUBLICATION BASED ON THE CURRENT RESEARCH

Published:

[1] **Paper 1**: Singh, B. & Srivastava, K. & Gupta, D.. (2019). Future prospects and challenges in geospatial database for handling of big data concept: A review. International Journal of Recent Technology and Engineering. 7. 140-144.

Weblink: https://www.ijrte.org/download/volume-7-issue-6c/

[2] **Paper 2:** Singh, B. & Srivastava, K. & Gupta, D.. (2020). Unique indexing model in Geospatial database Paradigm. Indian Journal of Computer Science and Engineering (IJCSE) Vol. 11 No. 2 Mar-Apr 2020 (204-216), e-ISSN : 0976-5166, p-ISSN : 2231-3850. DOI : 10.21817/indjcse/2020/v11i2/201102166

Weblink: https://www.ijcse.com/ijcse-issue.html?issue=20201102