DEVISING A FEATURE MATCHING ALGORITHM FOR PHARMACOPHORE MODELLING

Dissertation submitted to the Central University of Punjab

For the award of Degree of

Master of Technology

In

Computer Science & Technology

By

Suchismita Mahato

Supervisor

Prof. (Dr.) A. K. Jain

Co-Supervisor

Dr. Mahesh Kulharia

Centre for Computer Science & Technology

School of Engineering & Technology

Central University of Punjab

August, 2015

DECLARATION

I declare that the dissertation entitled "**DEVISING A FEATURE MATCHING ALGORITHM FOR PHARMACOPHORE MODELLING**" has been prepared by me under the guidance of **Prof. A. K. Jain**, Dean, Centre for Computer Science and Technology, School of Engineering and Technology and **Dr. Mahesh Kulharia**, Assistant Professor, Centre for Bioinformatics, School of Emerging Life Science and Technology, Central University of Punjab.

No part of this dissertation/thesis has formed the basis for the award of any degree or fellowship previously.

Suchismita Mahato
CUPB/MTECH/SET/CST/2013-14/18
Centre for Computer Science and Technology,
School of Engineering and Technology,
Central University of Punjab,
Bathinda – 151001.

Date:

CERTIFICATE

It is certified that Suchismita Mahato has prepared her dissertation entitled "**DEVISING A FEATURE MATCHING ALGORITHM FOR PHARMACOPHORE MODELLING**", for the award of M.Tech degree of the Central University of Punjab, under my guidance. She has carried out this work at the Research Laboratory, Centre for Computer Science and Technology, School of Engineering and Technology, Central University of Punjab.

Prof. (Dr.) A. K. Jain
Dean
School of Engineering and Technology
Central University of Punjab,
Bathinda - 151001.

Dr. Mahesh Kulharia
Assistant Professor
Centre for Bioinformatics,
School of Emerging Life Science and Technology,
Central University of Punjab,
Bathinda - 151001.

Date:

ABSTRACT

Devising a Feature Matching Algorithm for Pharmacophore Modelling

Name:	Suchismita Mahato
Registration number:	CUPB/MTech/SET/CST/2013-14/18
Degree for which submitted:	Master of Technology
Name of supervisor:	Prof.(Dr.) A. K. Jain
Name of co-supervisor:	Dr. Mahesh Kulharia
Name of the centre:	Computer Science & Technology
Name of the school:	School of Engineering & Technology

Keywords: Feature Matching, pharmacophore, SMILES, UCK.

Feature Matching is one of the central issues of model-based recognition and an important component of most object recognition systems. The feature matching technique can be applied in image alignment, 3D secure reconstruction, motion tracking, object recognition and in pharmacophore based molecular alignment. Even though numerous algorithms exist for feature recognition and search yet the efficiency of these have considerable room for improvement. This has necessitated a continuous search for better performing algorithms. The combination of geometric hashing technique and tolerance limits for a given set of known interactions can be used for the pharmacophore modelling based search. The algorithm proposed is precise as compared with the previous existing algorithms. This algorithm is better in its runtime complexity.

(Suchismita Mahato) (Dr. Mahesh Kulharia) Co-supervisor (Prof. A. K. Jain) Supervisor

DEDICATED TO MY PARENTS

&

THE BEAUTIFUL CONCEPT

OF

GEOMETRIC HASHING

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation to my supervisor **Prof. (Dr.) A. K. Jain**, COC Centre for Computer Science and Technology, for his counsel.

It is a genuine pleasure to express my deep sense of thanks and gratitude to my mentor, philosopher and guide **Dr. Mahesh Kulharia,** Assistant Professor at Central University of Punjab (CUP), Bathinda, who has guided and supported me throughout my thesis with his patience and knowledge whilst allowing me the room to work in my own way. I attribute the level of my Master's degree to his encouragement and effort and without him this thesis, too, would not have been completed. One simply could not wish for a better or friendlier co-supervisor. It was his strong perseverance and determination that led to the emergence of new algorithms in my dissertation work. I successfully overcame many difficulties and learned a lot. It is to him that I dedicate my thesis.

Let me place on record my gratitude to **Prof. P. Ramarao**, Dean-Academic Affairs, for giving me this opportunity to carry out my thesis work.

It is a privilege to thank my friend **Er. Vicky Kumar** for his helpful guidance and continuous support in my coding of the dissertation work. I express my deepest appreciation to my friend **Er. Saurav Kumar Jindal** without whose co-operation, patience in listening to my problems, solving them invariably and providing timely suggestions, my algorithms would not have been completed.

Lastly, I would like to pay high regards to my parents and friends for their endless support throughout my life.

(Suchismita Mahato)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Sr. No.	Word	Abbreviation
1.	Accuracy	ACC
2.	Computer-Aided Design	CAD
3.	Computer-Aided Manufacturing	CAM
4.	Chemical Abstracts Service	CAS
5.	Content Based Image Retrieval	CBIR
6.	Content-Based Visual Information Retrieval	CBVIR
7.	Crystallographic Information File	CIF
8.	Chemical Markup Language	CML
9.	Common Geometrical Pattern Search System	COMPASS
10.	Central Processing Unit	CPU
11.	Depth First Search	DFS
12.	European Molecular Biology Open Software Suite	EMBOSS
13.	Feature Point Pharmacophores	FEPOS
14.	False Negative	FN
15.	False Positive	FP
16.	False Positive Rate	FPR
17.	Genetic Algorithm Superposition Program	GASP
18.	Graphical User Interface	GUI
19.	Hyper Text Markup Language	HTML
20.	Integrated Development Environment	IDE
21.	International Union of Pure and Applied Chemistry	IUPAC
22.	Java Server Pages	JSP
23.	Java Virtual Machine	JVM
24.	Linux, Apache, MySQL, Perl/PHP/Python	LAMP

Sr. No.	Word	Abbreviation
25.	Lesser General Public Licence	LGPL
26.	Master Data Library	MDL
27.	Molecular Operating Environment	MOE
28.	National Cancer Institute	NCI
29.	Operating System	OS
30.	Protein Data Bank	PDB
31.	Pre-Processor Hypertext	PHP
32.	Query By Image Content	QBIC
33.	Quantitative Structure-Activity Relationship	QSAR
34.	Random Sampling Consensus	RANSAC
35.	Relational Database Management System	RDBMS
36.	Simplified Molecular-Input Line-Entry System	SMILES
37.	True Negative	TN
38.	True Positive	TP
39.	Universal Chemical Key	UCK
40.	Extended Markup Language	EML

CHAPTER 1
INTRODUCTION

A feature is used as a starting point for many computer vision algorithms. “Features” are generally highly application-dependent. A feature is reflected to signify the effect of an occurrence, which is conditioned on some check, which in turn depends on the input data stream. The fact that the check is satisfied means that the feature appears, and feature parameters can then be estimated accordingly from local data. Since the purpose of a feature generally necessitates that some purposeful be applied to the input data stream and that functional is typically reapplied at each distinct “location” in the input stream, the output of such a functional is occasionally called the “feature” information. The output information is a feature of the image in the previous sense, where the test is always true (Kheng, 2007).

When observed as a problem of matching collections of feature vectors, the following three sub-problems can be identified within the object recognition problem:

- Finding correspondences between extracted and model structures, after the model has been suitably transformed into the image domain;
- Calculating a score similarity measure based on those correspondences, by means of a measure for the distance between sets of features;
- Looking for the transformation that best superimposes a model view into the image to optimize the ultimate score.

Matching is one of the vital issues of model-based recognition and a significant component of most object recognition systems. A common goal is to project a three-dimensional model in a scene at roughly the correct position, with a similar scale and alike alignment to the object in the section. In that way, the precision of the match concerning the object model and the image data can be calculated and compared, and candidate hypotheses can be authenticated in a verification stage. Accomplishing this task necessitates that pertinent and comparable information be extracted from both the sensor data as well as the model database (Keller, 1999).

Given a Euclidean distance metric, the simplest matching strategy is to set a threshold (maximum distance) and to return all matches from other images

within this threshold. Setting the threshold too high results in too many false positives, i.e., incorrect matches being returned. Setting the threshold too little results in too many false negatives, i.e., too many correct matches being missed. We can quantify the performance of a matching algorithm at a particular threshold by first counting the number of true and false matches and match failures, using the following definitions

- TP: true positives, i.e., number of precise matches;
- FN: false negatives, matches that were not appropriately detected;
- FP: false positives, estimated matches that are improper;
- TN: true negatives, non-matches that were appropriately rejected.

Expressions:

1) True Positive Rate,

$$TPR = \frac{TP}{TP+FN} = \frac{TP}{P} \quad (1.1)$$

2) False Positive Rate,

$$FPR = \frac{FP}{FP+TN} = \frac{FP}{N} \quad (1.2)$$

3) Accuracy,

$$ACC = \frac{TP+TN}{P+N} \quad (1.3)$$

Once decided on a matching strategy, potential candidates are needed to be efficiently searched. The humblest way to find all equivalent feature points is to equate all features against all other features in each pair of hypothetically matching images. Unfortunately, this is quadratic in the number of extracted features, which makes it impractical for most applications (Kheng, 2007).

An enhanced approach is to develop an indexing structure such as a multi-dimensional search tree or a hash table to search quickly for features near a given feature. Such indexing structures can either be built for each image independently (which is useful if we want only to consider certain potential matches, e.g., searching for a particular object), or universally for all the images in a given database, which can potentially be faster, since it removes the need to iterate over each image. For enormously large databases (millions of images or more), even more proficient structures based on philosophies from document recovery (e.g., vocabulary trees) can be used.

One of the simpler techniques to implement is multi-dimensional hashing, which maps descriptors into fixed size buckets grounded on some function applied to each descriptor vector. At matching time, each new feature is hashed into a bucket, and a search of nearby buckets is used to return potential candidates.

1.1. Overview of the Dissertation

The feature matching technique can be applied in image alignment, 3D secure reconstruction, motion tracking, object recognition and in pharmacophore based molecular alignment. The problem of feature matching is to select a subset of points in a specific spatiotemporal domain and identify its features such that variation in the coordinates of the selected subset can be studied. The time-dependent variation in the feature should not obscure the ability of the tool to keep it in focus (as the area under study). Even though numerous algorithms exist for feature recognition and search, yet the efficiency of these have considerable room for improvement. This has necessitated a continuous search for better performing algorithms.

In this dissertation work, molecules are obtained from any of the Chemical databases (ZINC, ChemBridge) and a new numbering scheme is applied to renumber the atoms so that the proposed technique is independent of the numbering scheme already provided in the Structure file. The numbering scheme is very subjective, and there is no clear and reproducible mechanism of numbering of atoms, hence a new numbering scheme was developed. Subsequently, a unique path is generated which traces the path of the entire molecule. It is required to trace the path of the entire molecule to remove the redundancy which can occur in the later stage while the unique key is generated. This unique path is used to obtain the key of the molecule. Triangles are then generated from the coordinates of each point of the molecule. The points are nothing but the coordinates of the atoms of the molecule. Thereafter, the largest side, its tolerance limits, the adjacent angles and their tolerance limits are calculated for each of the triangles generated. This is done by using Geometric hashing. All these information are stored into the database with the table name being the unique key of the molecule obtained in the earlier stage. For the feature matching phase, any two molecules are selected from the database and these features (largest side, adjacent angles) are compared.

1.2. Applications of Feature Matching

Feature Matching can be applied in:

1.2.1. Image Alignment

Image registration is the method of converting dissimilar sets of data into one coordinate system. Data may be numerous photographs, data from different sensors, times, depths, or viewpoints. It is used in computer vision, medical imaging, biological imaging and brain mapping, military automatic target recognition, and compiling and analysing images and data from satellites.

1.2.2. 3D Secure Reconstruction

3D reconstruction from numerous images is the construction of three-dimensional models from a set of images. It is the opposite process of obtaining 2D images from 3D scenes. The nature of an image is a prognostication from a 3D scene onto a 2D plane, during which course the depth is lost. The 3D point equivalent to a definite image point is forced to be on the line of sight. From a sole image, it is impossible to regulate which point on this line corresponds to the image point. If two images are available, then the location of a 3D point can be established as the connexion of the two projection rays. This process is denoted as triangulation. The fundamental for this process is the relations between multiple views which convey the information that analogous sets of points must contain some structure. This structure is related to the poses and the calibration of the camera.

1.2.3. Motion Tracking

Match moving is a cinematic procedure that permits the insertion of computer graphics into live-action footage with correct position, scale, and motion relative to the photographed objects in the shot. The term is used to describe various different methods of extracting camera motion information from a motion picture. Match moving is associated to rotoscoping and photogrammetry, and it is sometimes referred to as motion tracking or camera solving. Match moving is occasionally muddled with motion capture which records the motion of objects, rather than the camera. Normally, motion capture requires special cameras and sensors and a controlled environment. Match moving is also distinguishable from motion control photography, which uses mechanical hardware to execute multiple identical camera moves. In contrast, match moving is characteristically a software-

based technology, applied after the fact to normal footage recorded in unrestrained situations with an ordinary camera. Match moving is primarily used to track the movement of a camera through a shot so that an indistinguishable virtual camera move can be reproduced in a 3D animation program. When new animated elements are composited back into the original live-action shot, they will appear in perfectly-matched perspective and, therefore, appear seamless.

1.2.4. Object Recognition

Object recognition is a computer technology associated to computer vision and image processing that deals with identifying illustrations of semantic objects of a definite class in digital images and videos. Algorithms of object recognition include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

1.2.5. Image Indexing and Content-Based Retrieval

Images can be indexed and extracted by textual descriptors and by image content. In textual queries, words are used to extract images or image sets, and in visual queries, images or image sets are retrieved by visual characteristics such as colour, texture and shape. Image retrieval strategies based on either text-based or content-based retrieval alone have their restrictions. Images are subject to a wide range of elucidations, and textual descriptions can only initiate to arrest the richness and complexity of the semantic content of a visual image. Human indexing of images is highly labour-intensive and limiting when large databases are involved. However, retrieval based on visual characteristics is computationally deep and has not yet reached the point where it can be efficiently used to frame knowledgeably subtle queries, especially for non-specialist users. Content-based image retrieval (CBIR) is also known as query by image content (QBIC). The application of computer vision procedures to the image extraction problem is Content-based visual information retrieval (CBVIR), that is, the problem of scrutinizing digital images in large. Content-based image retrieval is quite different from traditional concept-based approaches. Content-based means that the search examines the contents of the image rather than the metadata such as keywords, tags, or descriptions related with the image. The term "content" in this situation might refer to colours, shapes, textures, or any other information that can be

acquired from the image itself. CBIR is recommendable because searches that depend purely on metadata are reliant on annotation quality and wholeness. Having humans manually interpret images by entering keywords or metadata in a large database can be time. The assessment of the effectiveness of keyword image search is subjective and has not been well-defined. Similarly, CBIR systems have identical challenges in defining success.

1.2.6. Pharmacophore

A pharmacophore can be defined as the group of atoms in the molecule of a drug responsible for the drug's action. This area is covered in detail in the next section of the report.

1.3. Pharmacophore

A pharmacophore model is a geometrical description of the chemical functionalities required of a ligand to interact with the receptor. The objective of computer-aided molecular design methods in contemporary medicinal chemistry is to lessen the overall cost allied with the discovery and development of a new drug by identifying the most promising candidates to emphasise on the experimental efforts.

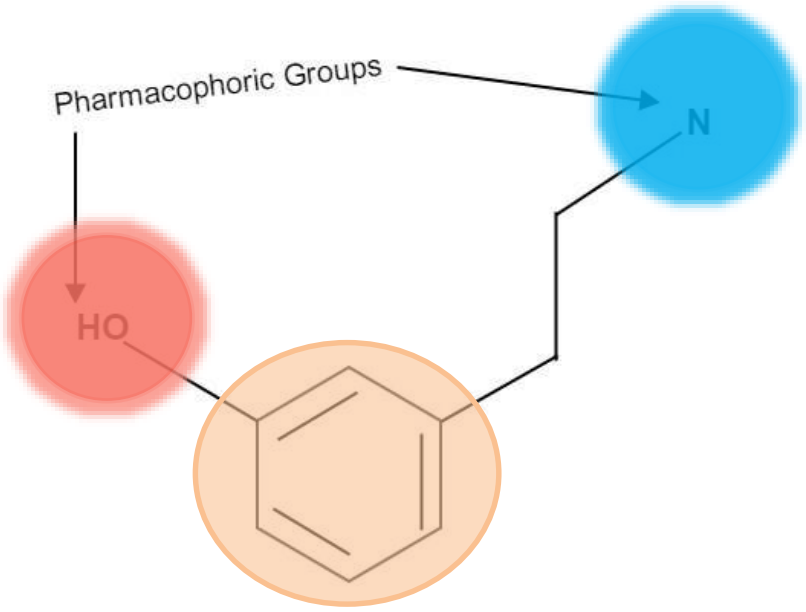


Figure 1.1: Diagram showing Pharmacophoric groups

Often, drug discovery projects have already reached a well-advanced stage before detailed structural data on the target has become available. Investigational screening for lead structure deduction is limited by the potential number of

compounds that succumb to a high-throughput bio-assay and with the low number of active drug-like compounds identified (<0.1%). The pharmacophore approach has proven to be accomplished within this context, allowing

i. the observation and understanding of key interactions between a target and a ligand and
ii. the improvement of hit rates acquired in experimental screening of subsets that have been obtained from silicon screening experiments.

A pharmacophore is the collection of these properties that form a vital part of a drug. A Pharmacophore may be defined as the necessary geometric ordering of atoms or functional groups necessary to construct a certain biological response. The strict IUPAC description of a pharmacophore is: “A pharmacophore is the ensemble of steric and electronic features that is necessary to ensure the optimal supramolecular interactions with a specific biological target structure and to trigger (or to block) its biological response” (Drie, 2010). Pharmacophore mapping is one of the significant elements of drug design in the deprivation of structural data of the target receptor. The tool initially applied to detection of lead molecules now expands to lead optimization. Pharmacophore can be used as queries for recovering potential leads from structural databases (lead discovery), for calculating molecules with fixed desired features (lead optimization), and for evaluating similarity and variety of molecules using pharmacophore fingerprints. Pharmacophore can also be applied in aligning molecules based on the 3D alignment of chemical features or to develop predictive 3D QSAR models (Kier, 2007).

The features need to match dissimilar chemical groups with similar properties, to spot novel ligands. Ligand-receptor interactions are characteristically “polar positive”, “polar negative” or “hydrophobic”. A well-defined pharmacophore model incorporates both hydrophobic volumes and hydrogen bond vectors.

I. ***Process of Pharmacophore:*** The following steps are involved in the process for developing a pharmacophore model:

i. Select a set of molecules: A set of molecules are selected that will be used for developing the pharmacophore model. The set of molecules should contain both active and inactive compounds, as a

pharmacophore model should be able to differentiate between molecules with and without bioactivity.

ii. Conformational analysis: Conformational analysis of the individual molecules is performed and a presumed bioactive conformation for each molecule is selected. The proposed bioactive conformation of a molecule should be a low-energy conformation for a pharmacophore model to be valid.

iii. Molecular superimposition: The low-energy conformations of the molecules are superimposed. Similar functional groups shared with all molecules in the set might be fitted (e.g., phenyl rings or carboxylic acid groups).

iv. Abstract description: The superimposed molecules are transformed into an abstract representation. For example, superimposed phenyl rings might be referred to as an 'aromatic ring' pharmacophore element. Similarly, hydroxyl groups could be elected as a 'hydrogen-bond donor/acceptor' pharmacophore element.

v. Validation: A pharmacophore model is a hypothesis investigating for the observed biological activities of a set of molecules. The model is only effective insofar as it can account for differences in the biological activity of a range of molecules.

Pharmacophore perspectives have become one of the major tools in drug discovery after the past century's development. Various ligand-based and structure-based methods have been evolved for improved pharmacophore modelling and have been successfully and expansively applied in virtual screening, de novo design and lead optimization. Despite these accomplishments, pharmacophore methods have not touched their expected full capacity, particularly in facing the demand for reducing the current expensive complete cost coupled with drug discovery and development (Ebalunode *et al*., 2008).

II. ***Challenges in Pharmacophore Modelling:*** The first challenging drawback is the modelling of ligand flexibility. Currently, two procedures have been used to deal with this problem: the first is the pre-enumerating method. In this method, multiple conformations for each molecule are pre-computed and saved in a database. The second is the on-the-fly method. In this way, the structure

analysis is carried out in the pharmacophore modelling process. The pre-enumerating method requires less computing cost, and the on-the-fly method requires less storage space but high CPU time.

The second perplexing issue in ligand-based pharmacophore modelling is an organisation of molecules. The alignment methods can be classified into two categories in terms of their fundamental nature. The first category is the point-based approach and the second approach is the property-based approach. The points can be further differentiated as atoms, fragments or chemical features in the point-based method. Pairs of atoms, chemical feature points or fragments are frequently covered using a least-squares fitting in point-based algorithms, The biggest restriction of these approaches is the requirement of predefined anchor points because the generation of these points can become troublesome in the case of dissimilar ligands. The property-based algorithms are, usually represented by sets of Gaussian functions and make use of molecular field descriptors to generate alignments.

The third challenging drawback lies in the determination of training set compounds. The points affecting shake the final generated pharmacophore models are the size of the dataset and its chemical diversity noticeably. In some cases, entirely different pharmacophore models of ligands interacting with the same macromolecular target could be created from the same algorithm and program that uses different training sets.

1.4. Introduction to the Pharmacophore Modelling

1) The problem of feature matching is to select a subset of points in a specific spatiotemporal domain and identify its features such that variation in the coordinates of the selected subset can be studied. The time-dependent variation in the feature should not obscure the ability of the tool to keep it in focus (as the area under study).
2) The feature matching technique can be applied in image alignment, 3D secure reconstruction, motion tracking, object recognition and in pharmacophore based molecular alignment.
3) Even though numerous algorithms exist for feature recognition and search, yet the efficiency of these have considerable room for improvement. This has

necessitated a continuous search for better performing algorithms. Some of the “better” performing algorithms are as under:

A. ***Holistic Matching Algorithms:*** Holistic matching algorithms mainly extract global features from the entire face. Eigen-faces and Fisher-faces are well-known examples of holistic face recognition algorithms. The limitation of holistic matching is that it requires accurate normalization of the faces according to pose, illumination and scale. Alterations in these factors can affect the global features taken out from the faces leading to inaccuracies in the terminal recognition. Normalization is usually carried out by manually recognizing landmarks on the faces which make the whole process semi-automatic. Manual normalization is also inaccurate as it is frequently performed on the basis of only a few standards (3 to 5), and humans cannot identify landmarks with sub-pixel accuracy. Restoring this manual process with an automatic landmark identification algorithm usually decreases the final recognition outcomes, and global features are also sensitive to facial expressions and occlusions.

   Feature-based matching algorithms have an advantage over holistic matching algorithms because they are robust to variations in pose, illumination, scale, expressions and occlusions.

B. ***Key-point Detection and Local Feature Matching for Textured 3D Face Recognition:*** A key-point detection technique repeatedly identifies key points at locations where shape variation is high in 3D faces. A distinctive 3D coordinate basis lubricates taking out of highly descriptive pose unchanged features. A 3D feature is taken out by fitting a surface to the neighbourhood of a key-point and sampling it on a uniform network. Features from a scrutiny and gallery face are projected to the subspace and equalled. Two graphs are constructed for the set of matching features. The resemblance between two faces is measured as the resemblance between their graphs. The aim of key-point detection is to deduce points on a surface which can be specified with high repeatability in different variation of images of the same surface in the presence of noise and pose variations (Lowe, 2004).

The disadvantages of the key-point detection algorithm are that the time complexity and space complexity is too high, and it does not give accurate results.

C. Feature Matching Using RANSAC: RANSAC stands for RANdom SAmple Consensus. The algorithm for RANSAC is as follows:

i. Randomly select a seed group of matches.
ii. Find inliers (distinct area) to this transformation.
iii. If the number of inliers is adequately huge, re-calculate least-squares assessment of transformation on all of the inliers.
iv. Keep the transformation with the highest number of inliers. (Chiu *et al.*, 2012).

A disadvantage of RANSAC is that there is no upper bound on the time it takes to compute these parameters. Moreover, RANSAC is not every time able to find the best set even for abstemiously contaminated sets, and it usually performs gravely when the quantity of inliers is less than 50. Another disadvantage of RANSAC is that it necessitates the setting of problem-particular thresholds. RANSAC can only estimate one model for a specific data set. As for any one model method when two (or more) model instances exist, RANSAC may fail to find either one.

4) The desired role of the proposed algorithm is that it would be error-tolerant. This algorithm would also be faster according to its running time complexity. The algorithm proposed would also be more precise as compared with the previous existing algorithms.

1.5. Geometric Hashing

Geometric hashing is a technique originally developed in computer vision for matching geometric features against a database. It finds use in some other areas. Matching is likely even when the identifiable database objects have undergone conversions or when only partial evidence is present. The technique is highly proficient and of low polynomial complexity.

The geometric hashing method allows systems to recognize objects even when the objects have experienced a capricious transformation and when parts of them might be obstructed. The speciality of the method is in its efficiency, in its

ability to operate in the existence of only partial information and in its applicability to almost any domain where geometric matching is required. It does not require any domain-specific knowledge, only the location of certain geometric interest features. The method has been effectively applied to pattern matching problems in computer vision, CAD/CAM and medical imaging.

Object recognition is the ultimate goal of most computer vision research. An ideal object recognition system should be able to recognize objects in an image that are partially obstructed or have undergone geometric transformations. Most systems will use a large database of models and apply model-based recognition.

For example, if the ability to recognize all objects and tools on a factory floor is to be given to a robot. If there are only a scarce hundred items, a database of these objects can be designed and stored in the robot's memory. When the robot obtains a sensory image of its environment from a video camera or a range sensor, it should be able to retrieve rapidly from memory objects that appear in the image. Although quite natural in human vision, this task in a robot requires the solution of several complicated problems such as

a. The objects in the attained scene appeared rotated and translated about their initial database position, and the whole scene undergoes a sensor dependent transformation, such as the projective transformation of a video camera.
b. The objects in the scene may partially obstruct each other, and the scene may include supplementary objects not listed in the database.
c. It is computationally inefficient to retrieve each distinct object from the database and associate it in contradiction of the observed scene in search of a match. For example, if the section contains only round objects, it does not make sense to retrieve rectangular objects to match against it.

A method is required that allows direct access to only the relevant information such as an indexing based approach. For example, if words in long strings of text are looked for, a table can be used accessed by indices that are functions of individual words. The table contains the strings where the word appears and the location of the word in the strings. It would be easy then to discover a word by retrieving all of its arrivals from the table. Another example could be finding the number of possible triangles from the number of points submitted as an input, finding the longest adjacent side of each triangle, both the

angles of the longest adjacent side and saving all the information and calculation into the database.

1.6. Problem Statement

The problem of feature matching is to select a subset of points in a specific spatiotemporal domain and identify its features such that variation in the coordinates of the selected subset can be studied. The time-dependent variation in the feature should not obscure the ability of the tool to keep it in focus (as the area under study). Although some feature matching algorithms have come up in recent years to identify pharmacophore based search, yet the optimal tool has remained elusive.

1.7. Objectives

1. To generate the tools for
    a) Detection of degenerate points.
    b) Reduction of points by Triangulation Theorem
    c) Generation of triangles and calculation of their features.

    To achieve this objective, the following sub-objectives need to be fulfilled:
    i. From a non-redundant database of protein-protein interaction interfaces, the information about the spatial orientation of interacting sub-molecular features shall be derived.
    ii. The tolerance limit for each of individual interactions will be calculated from the distribution of the spatio/geometric parameters.
2. Incorporation of tolerance limit in above mentioned tool.
3. Testing and validation of the tool.

1.8. Outline of the Dissertation

This dissertation unfolds in five chapters. Chapter one gives the brief introduction of feature matching and how the feature can be detected and matched. It also presents the applications of feature matching in various fields. One of its application lies in pharmacophore modelling. There can be a technique in which the molecules of a pharmacophore can be matched taking out its essential features. The description of the pharmacophore and its challenges are also covered in the introduction section.

It is followed by chapter two as it describes the related research work that has been done in the field of feature selection and pharmacophore by different researchers. This chapter begins by explaining the studies that have been done by various researchers in this area. Chapter three describes the various software tools which were used for the completion of the dissertation work. It also includes the proposed technique, and the methodology followed to fulfil my proposed work. Chapter four illustrates in detail the results obtained at every step of the dissertation. It also explains and summarises the results in a few lines. The experimental results have been analyzed at the end of this chapter.

Chapter six provides the conclusion and future work in this area of the research field. All the results and discussions have been concluded in this chapter in summarized form.

CHAPTER 2
REVIEW OF LITERATURE

Drie, 2010 recently introduced pharmacophore-based 3D searching tools, a simple tool with a limited language for expressing geometric relationships and a command-line user interface. That tool is being integrated into the 3D graphical-user-interface of Bioclipse. Conformational analysis tools are also slowly appearing. Balloon is not open source but is freely available in a binary format to calculate multiple conformers per molecule, in a relatively high-quality way. RDKIT.ORG is true open source and provides additional tools for multiple conformer generations. That was used in 2008 to compute the world's largest multi-conformer 3D database, consisting of the world's collection of commercially available compounds, as recorded by emolecules.com (http://www.emolecules.com/doc/plus/download-database.php), about 5 million compounds which are available as an open resource to the virtual screening community. Pharmacophore.org was created in 2008, primarily as a set of pointers to these open-source resources for those interested in pharmacophore searching and discovery.

Grossman *et al.*, 2008 developed an algorithm called the Universal Chemical Key (UCK) algorithm that builds a unique key for a molecular structure. The molecular structures are characterized as undirected labeled graphs with the atoms representing the vertices of the graph and the bonds representing the edges.

Sun, 2008 presented a paper in which BioSolveIT introduced the LeadIT software, which includes a component ReCore to perform 3D database searching, either with a user-defined pharmacophore or using CAVEAT-like vectorial relationships. In 2006, the computational chemistry software vendor Schrodinger introduced a novel pharmacophore discovery tool, PHASE, evidently aiming to mimic in functionality Catalyst's Hypothesis Generation, for automated pharmacophore discovery.

Kevin, 2008 presented a paper in which debates the effectiveness and matchlessness of SMILES algorithm show how the SMILES algorithm fails to produce distinctive identifiers and show how they can adjust from the SMILES algorithm to produce identifiers that are unique and useful. It also developed an algorithm named Graph Linear Representer (GLR) that constructs a unique

identifier for chemical structures. This algorithm fails to identify multiple bonds, and it works on two-dimensional structures only.

Jenkins *et al.*, 2004 used GASP, A notable, creative approach, which used molecular field similarity in conjunction with an evolutionary algorithm to find optimal overlays. Val Gillett in Sheffield continues to probe novel approaches to pharmacophore discovery. Finn and Srinivasan had an interesting new take to the pharmacophore discovery problem which emerged under the title of Inductive Logic Programming of Doug Rohrer, JimPetke and Gerry Maggiora developed MIMIC, an independent but intellectually similar method to GASP which arose initially in Upjohn Labs and was extended into a tool for virtual screening by Jordi Mestres, and put into fruitful application by him at Organon. MIMIC, like GASP, relied on field similarity for its superpositioning. FEPOPS (feature-point pharmacophores) was published and patented at Novartis by Jeremy Jenkins et al., which uses pharmacophore-like descriptors to perform virtual screening and to analyze high-throughput screening data.

Steindl & Langer, 2004 presented that in the first decade of the new millennium, Austria became a hub of interest in pharmacophore methodology and application, sparked by the enthusiasm of Thierry Langer. While Langer's initial applications centred on the use of Catalyst, he founded InteLigand in 2003 with Gerhard Wolberand Hermann Stuppner centered on novel tools of their own design, ones which provided both pharmacophore 3D database searching, and pharmacophore discovery. MIMIC, like GASP, relied on field similarity for its superpositioning.

Jenkins *et al.*, 2004 patented the FEPOPS (feature-point pharmacophores) which uses pharmacophore-like descriptors to perform virtual screening and to analyze high-throughput screening data. He developed a software suite named DANTE hewing closely to the protocol inherent in Catalyst while attempting to solve Catalyst's numerous shortcomings; the algorithms were published but never commercialized. A key advance was the introduction of the 'shrink-wrap' algorithm to determine the steric boundaries of the receptor site, without prior knowledge of the bioactive conformation, something we often attempted in Catalyst, without success.

Jain, 1994 introduced COMPASS, which attempted to find an optimal overlay of all active molecules, but innovatively did not demand that all features overlay in a

common way, optimizing bioactive conformation and pose simultaneously. Attention then focused more on the application of these methods to real problems in drug discovery, rather than methodological innovation.

Gasteiger *et al,* 1992 presented that in the early- to mid-1990s, CORINA appeared from Molecular Networks, a startup spun out of Gasteiger's group at Erlangen in Germany. It produced a single high-quality conformation. OMEGA appeared in the mid-1990s, from Open Eye Scientific Software, and provided the capability to produce multiple conformations, at a speed suitable for 3D database construction. In the mid-1990s, Chemical Computing Group's MOE software was enhanced to include pharmacophore searching and pharmacophore discovery. Uniquely, they use some type of feature mapping which places features floating above the atoms to which they are mapped, which often results in unintuitive behaviour.

Pharmacophore-based 3D database searching was explicitly invented for making prospective predictions based on a pharmacophore model, by searching a database of conformations of molecules for which samples are readily available for biological testing. Many virtual hits show no activity, but even if 1% of the hits are active, the exercise will probably be useful to drug discovery teams, as the cost of assaying existing compounds is usually low. It is not uncommon to see hit rates nearer to 10%. While searching 3D databases using a pharmacophore originated in the work of Peter Gund in the late 1970s, the development of novel approaches in the late 1980s and their role in the first successful virtual screens brought these techniques into prominence; simultaneous to that, these tools moved from the realm of academic research and proprietary corporate research into the commercial realm, notably ALADDIN and MACCS-3D, which encouraged their broader use and adoption.

A necessary precursor to searching a 3D database using a pharmacophore is the process of distilling a pharmacophore from an SAR (structure–activity relationship), a process variously called pharmacophore identification, pharmacophore elucidation, or pharmacophore discovery. This work also had roots in academic labs from the late 1970s, followed by key contributions from proprietary corporate research; APOLLO was very nearly the first automated pharmacophore discovery, from Konrad Koehler and Jim Snyder. APOLLO lacked

the automated determination of which features were important. The first software to perform this process in a fully automated way was Catalyst (created by the now-defunct BioCAD Corporation, absorbed by Accelrys in 1994), whose introduction in the early 1990s was followed many related efforts, notably DISCO and GASP.

An utterly distinct line of evolution was initiated by Crippen in the late 1970s in **Boulo & Crippen, 1989**, which led to the subfield labelled 'distance geometry'. This line was pursued by several investigators and culminated in Crippen's work on 'Voronoi polyhedra' – attempts to infer the shape of a binding site based on SAR. The primary legacy of these efforts in today's methods is the role of distance geometry in the exhaustive conformational analysis.

Weininger *et al.*, 1989, Designed the chemical notation language SMILES for the conversion of an arbitrarily chosen description of a chemical structure to one unique notation. This is accomplished in a two-stage algorithm, CANGEN. The first stage involves CANonicalization of structure, whereby the molecule is treated as a graph with nodes (atoms) and edges (bonds). Each atom is canonically ordered and labeled. In the second stage, starting with the lowest labeled atom, a molecular graph is GENerated, which is the unique SMILES structure.

Marshall *et al.*, 1979, co-founded the Journal of Computer-Aided Molecular Design, which in its early days hosted a preponderance of manuscripts on pharmacophore-related topics. In particular, one of the most important publications on the methodology for automated pharmacophore determination appeared as the very first paper in the first very issue of Marshall's Journal of CAMD, 'A unique geometry of the active site of angiotensin-converting enzyme steady with structure–activity studies'. The most important contribution of Marshall and his coworkers to the concept of a pharmacophore was to expand this notion to include the steric boundaries of the putative site, in addition to the spatial arrangement of functional groups. They developed simple approaches for computing these boundaries by subtracting the volume enclosed by all the actives from the volume enclosing all the inactive. This is surprisingly effective, when either the molecules are rigid, or the bioactive conformation is known.

Tarjan, 1971, presented a paper on Depth First Search. The value of depth-first search or "backtracking" as a technique for solving problems is illustrated by two examples. An improved version of an algorithm for finding the strongly connected

components of a directed graph and ar algorithm for finding the biconnected components of an undirect graph are presented. The space and time requirements of both algorithms are bounded by k1V + k2E+ k_3 for some constants k1, k2, and k3, where Vis the number of vertices and E is the number of edges of the graph being examined.

2.1 Summary

The concept of the pharmacophore originated with Monty Kier in a series of papers 1968–1971 in **Kier, 2007**. He enunciated the theory of a pharmacophore and laid out its role in the methodology of 'receptor mapping'. In fact, not only did he introduce the term and calculate the first pharmacophore. Kier's work explicitly inspired several followers. In 1977, Gund was the first to link up a pharmacophore to a search engine, searching a database of 3D conformations. Those 3D conformations were crystallographically determined structures. Gund's work in turn inspired efforts by Brint and Willett, work that took place in the mid-1980s. The review ends with today's emergence of open source tools for pharmacophore searching. Even though numerous algorithms exist for feature recognition and search yet the efficiency of these have considerable room for improvement. This has necessitated a continuous search for better performing algorithms.

CHAPTER 3
MATERIAL AND METHODS

3.1. System Usage

3.1.1. Software Usage

- **Operating System:** Ubuntu Linux (Open Source)
- **Software:** JDK 7, Net Beans/Eclipse
- **Programming Language:** Java
- **Tools for statistical analysis:** R/MATLAB/Gnuplot

3.1.2. Hardware Usage

- **Processor:** >2.6GHz x86 processor (i5 or i7)
- **RAM:** >12 GB of system memory (RAM)
- **Hard Disk:** >500 GB of disk space

3.2. Software Tools

3.2.1. NetBeans

NetBeans is an Integrated Development Environment (IDE) for evolving principally with Java, but also with other languages, in particular, C/C++, PHP, and HTML5. It is also an application platform framework for Java desktop applications and others. The NetBeans IDE can run on Windows, OS X, Linux, Solaris and other platforms supporting a compatible JVM is written in Java.

The NetBeans Platform permits applications to be established from a set of sectional software components called modules. Applications based on the NetBeans Platform (counting the NetBeans IDE itself) can be stretched by third party developers. The NetBeans Team keenly supports the product and seek feature suggestions from the broader community. Every release is preceded by a time for Community testing and feedback.

The NetBeans Editor indents lines, matches brackets and words, and highlights source code semantically and syntactically. It also delivers code templates, coding tips and refactoring tools. The editor supports many languages from Java, C/C++, Groovy, Javadoc, XML and HTML, to PHP, JavaScript and JSP.

The rate of buggy code rises the longer it remains unfixed. NetBeans provides static analysis tools, especially integration with the broadly used FindBugs tool, for recognising and fixing common hitches in Java code.

3.2.2. BioJava

BioJava is an open-source project devoted to providing a Java framework for handling biological data. It provides statistical and analytical routines, parsers for common file formats and permits the handling of sequences and 3D structures. The objective of the biojava project is to enable rapid application development for bioinformatics.

BioJava is certified under LGPL 2.1. The BioJava project grew out of labour by Thomas Down and Matthew Pocock to form an API to streamline development of Java based Bioinformatics tools. BioJava is one of a number of Bio* projects designed to reduce code duplication. Examples of such tasks that fall under Bio* apart from BioJava are BioPython, BioPerl, BioRuby, EMBOSS, etc.

BioJava delivers software modules for many of the characteristic tasks of bioinformatics programming. These include:

- Accessing nucleotide and peptide sequence data from local and remote databases
- Transforming formats of database/ file records
- Protein structure parsing and manipulation
- Manipulating individual sequences
- Searching for similar sequences
- Creating and manipulating sequence alignments

3.2.3. Jmol

Jmol is an open-source Java viewer for chemical structures in 3D that does not involve 3D acceleration plugins. Jmol yields a 3D demonstration of a molecule that may be used as a teaching tool, or for study e.g. in biochemistry and chemistry. It is free and open source software, written in Java and so it runs on Windows, Linux and Unix systems and Mac OS X. There is a separate application and a development toolkit that can be assimilated into other Java applications, such as Bioclipse and Taverna.

A common feature is an applet that can be assimilated into web pages to spectacle molecules in a multiplicity of ways. For example, molecules can be exhibited as "ball and stick" models, "space filling" models, "ribbon" models, etc. Jmol funds an extensive range of molecular file formats, including MDL Molfile (mol), Chemical Markup Language (CML), Protein Data Bank (pdb), and Crystallographic Information File (cif).

The Jmol applet, among other skills, offers a substitute to the Chime plugin, which is no longer under active development. While Jmol has many features that are not available in Chime, it does not title to replicate all Chime functionality (most notably, Chime's Sculpt mode). Chime necessitates plug-in installation and Internet Explorer 6.0 or Firefox 2.0 on Microsoft Windows, or Netscape Communicator 4.8 on the MacintoshOS9. Jmol requires Java installation and operates on a wide variety of platforms. For example, Jmol is fully efficient in Mozilla Firefox, Internet Explorer, Opera, Google Chrome and Safari.

3.2.4. MySql

MySQL is the world's second most widely used relational database management system (RDBMS) and most widely used open-source RDBMS.

The SQL abbreviation stands for Structured Query Language. MySQL is a popular choice of database for use in web applications and is a vital constituent of the broadly used LAMP open source web application software stack (and other 'AMP' stacks). LAMP is an abbreviation for "Linux, Apache, MySQL, Perl/PHP/Python." Free software-open source projects that necessitate a full-featured database management system often use MySQL.

MySQL is a relational database management system (RDBMS). It ships with no GUI tools to direct MySQL databases or achieve data enclosed within the databases. Users may use the encompassed command line tools, or use MySQL "front-ends", desktop software and web applications that generate and achieve MySQL databases, build database structures, back up data, inspect status, and work with data records. The authorized set of MySQL front-end tools, MySQL Workbench, is actively developed by Oracle, and is easily accessible for use.

How MySQL works?

1. Connecting to and Disconnecting from the Server

A MySQL username and a password are required to connect to a server. If the server runs on a machine other than the one where it is logged in, a host name also needs to be specified. An example of how MySQL is started is shown in the snapshot below. Here, in this example it can be seen that the password for root is not mandatory.

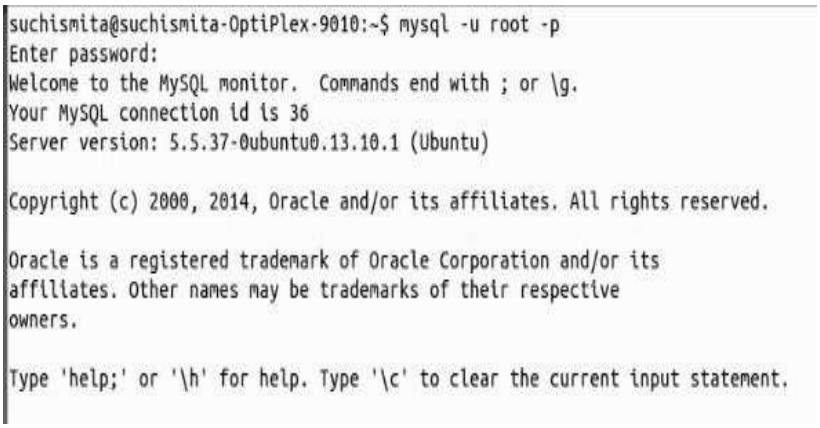


Figure 3.1: Connecting to the MySQL Server

After the password has been entered, some introductory information followed by a mysql> prompt is displayed as shown in the figure above. There is no password in this snapshot.

After it has been connected successfully, it can be disconnected any time by typing EXIT at the mysql> prompt:

Figure 3.2: Exiting from MySQL

2. Creating and Using a Database

A database is used to keep the track of various records with the help of tables. A collection of tables can be considered to be as a database. There are various operations which can be performed on the database. Some of them are as follows:

i. Create a database

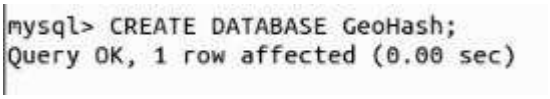


Figure 3.3: Creating a database

```
mysql> use GeoHash;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
```

Figure 3.4: Using a database

ii. Create a table

```
mysql> CREATE TABLE demo(SNO int not null, name varchar(25));
Query OK, 0 rows affected (0.08 sec)
```

Figure 3.5: Creating a table

iii. Load data into the table

```
mysql> insert into demo values(1, "Suchismita");
Query OK, 1 row affected (0.04 sec)

mysql> insert into demo values(2, "Rahul");
Query OK, 1 row affected (0.27 sec)

mysql> insert into demo values(3, "Manpreet");
Query OK, 1 row affected (0.42 sec)
```

Figure 3.6: Load data into the table

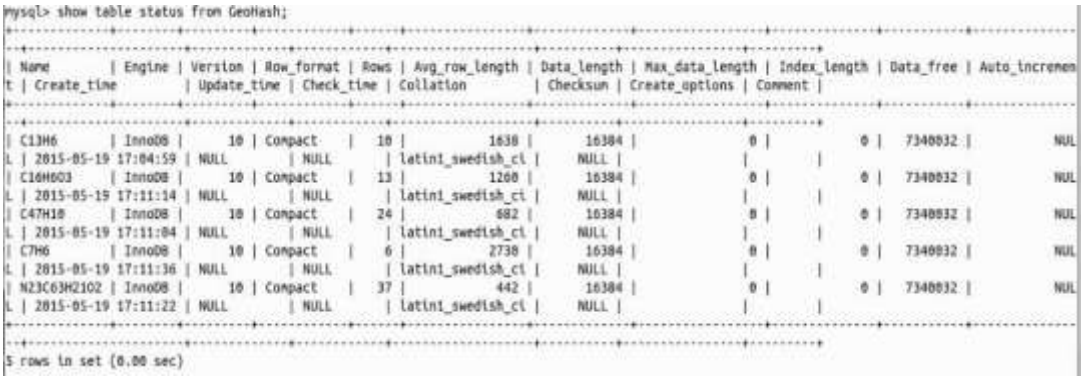
iv. Retrieve data from the table

```
mysql> select * from C13H6;
+------+---------+---------+----------+------------+---------+---------+----------+------------+---------+---------+----------+------------+---+
| SNo  | X1      | Y1      | Z1       | ATOM_NAME1 | X2      | Y2      | Z2       | ATOM_NAME2 | X3      | Y3      | Z3       | ATOM_NAME3 | L |
ARGEST | TOL1    | TOL2    | ANGLE1   | ANG1_LIMIT | ANGLE2  | ANG2_LIMIT | PATH                |
+------+---------+---------+----------+------------+---------+---------+----------+------------+---------+---------+----------+------------+---+
|    1 | 4.66000 | 0.79100 | -8.91900 | C          | 3.61600 | 0.77600 | -9.19600 | H          | 5.27400 |  1.99900 | -8.56000 | C          | 2
.15385 | 0.07950 | 0.10500 | 34.24697 |    4.66742 | 25.73955 |    3.86954 | CHCCHCCHCCHCCCHCCHC |
|    2 | 3.61600 | 0.77600 | -9.19600 | H          | 5.27400 | 1.99900 | -8.56000 | C          | 4.70400 | 2.91600 | -8.57300 | H          | 2
.48022 | 0.07950 | 0.07950 | 59.99692 |    2.80378 | 25.72971 |    2.51573 | CHCCHCCHCCHCCCHCCHC |
|    3 | 5.27400 |  1.99900 | -8.56000 | C          | 4.70400 | 2.91600 | -8.57300 | H          | 0.62700 |  2.01800 | -8.20900 | C          | 2
.15333 | 0.07950 | 0.10500 | 34.26904 |    4.66836 | 25.73968 |    3.86915 | CHCCHCCHCCHCCCHCCHC |
|    4 | 4.70400 |  2.91600 | -8.57300 | H          | 0.62700 |  2.01800 | -8.20900 | C          | 7.10100 |  2.95000 | -7.93800 | H          | 2
.47992 | 0.07950 | 0.07950 | 59.98518 |    2.80364 | 25.74411 |    2.51604 | CHCCHCCHCCHCCCHCCHC |
|    5 | 0.62700 |  2.01800 | -8.20900 | C          | 7.10100 |  2.95000 | -7.93800 | H          | 7.36600 |  0.82900 | -8.20200 | C          | 2
.15373 | 0.07950 | 0.10500 | 34.26086 |    4.66697 | 25.74459 |    3.86895 | CHCCHCCHCCHCCCHCCHC |
|    6 | 7.10100 |  2.95000 | -7.93800 | H          | 7.36600 |  0.82900 | -8.20200 | C          | 8.41000 |  0.84400 | -7.92600 | H          | 2
.47969 | 0.07950 | 0.07950 | 60.01670 |    2.80148 | 25.74295 |    2.51559 | CHCCHCCHCCHCCCHCCHC |
|    7 | 7.36600 |  0.82900 | -8.20200 | C          | 8.41000 |  0.84400 | -7.92600 | H          | 5.39900 | -0.39800 | -8.91200 | C          | 3
.40307 | 0.07950 | 0.10500 | 20.86295 |    5.19723 |  9.12736 |    2.44046 | CHCCHCCHCCHCCCHCCHC |
|    8 | 8.41000 |  0.84400 | -7.92600 | H          | 5.39900 | -0.39800 | -8.91200 | C          | 4.92500 | -1.33000 | -9.18300 | H          | 4
.29553 | 0.07950 | 0.07950 | 29.98815 |    4.82398 |  9.12835 |    1.74495 | CHCCHCCHCCHCCCHCCHC |
|    9 | 5.39900 | -0.39800 | -8.91200 | C          | 4.92500 | -1.33000 | -9.18300 | H          | 6.75200 | -0.37900 | -8.55400 | C          | 2
.15359 | 0.07950 | 0.10500 | 34.25187 |    4.66745 | 25.74356 |    3.86973 | CHCCHCCHCCHCCCHCCHC |
|   10 | 4.92500 | -1.33000 | -9.18300 | H          | 6.75200 | -0.37900 | -8.55400 | C          | 7.32200 | -1.29600 | -8.54800 | H          | 2
.47992 | 0.07950 | 0.07950 | 59.99761 |    2.80371 | 25.73332 |    2.51596 | CHCCHCCHCCHCCCHCCHC |
+------+---------+---------+----------+------------+---------+---------+----------+------------+---------+---------+----------+------------+---+
10 rows in set (0.00 sec)
```

Figure 3.7: Retrieve data from the table using SELECT command

3. Getting Information about databases and tables

To figure out what tables the default database contains (for example, when the table name is unknown), use this command:



```
mysql> show table status from GeoHash;
+------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+-------------+------------+-----------------+----------+----------------+---------+
| Name       | Engine | Version | Row_format | Rows | Avg_row_length | Data_length | Max_data_length | Index_length | Data_free | Auto_increment | Create_time         | Update_time | Check_time | Collation       | Checksum | Create_options | Comment |
+------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+-------------+------------+-----------------+----------+----------------+---------+
| C13H6      | InnoDB |      10 | Compact    |   10 |           1638 |       16384 |               0 |            0 |   7340032 |           NULL | 2015-05-19 17:04:59 | NULL        | NULL       | latin1_swedish_ci |     NULL |                |         |
| C16H6O3    | InnoDB |      10 | Compact    |   13 |           1260 |       16384 |               0 |            0 |   7340032 |           NULL | 2015-05-19 17:11:14 | NULL        | NULL       | latin1_swedish_ci |     NULL |                |         |
| C47H18     | InnoDB |      10 | Compact    |   24 |            682 |       16384 |               0 |            0 |   7340032 |           NULL | 2015-05-19 17:11:04 | NULL        | NULL       | latin1_swedish_ci |     NULL |                |         |
| C7H6       | InnoDB |      10 | Compact    |    6 |           2730 |       16384 |               0 |            0 |   7340032 |           NULL | 2015-05-19 17:11:36 | NULL        | NULL       | latin1_swedish_ci |     NULL |                |         |
| N23C63H210Z | InnoDB |      10 | Compact    |   37 |            442 |       16384 |               0 |            0 |   7340032 |           NULL | 2015-05-19 17:11:22 | NULL        | NULL       | latin1_swedish_ci |     NULL |                |         |
+------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+-------------+------------+-----------------+----------+----------------+---------+
5 rows in set (0.00 sec)
```

Figure 3.8: Getting information about the database

If the structure of a table is needed to be determined, the DESCRIBE statement is suitable; it shows information about each of a table's columns:

```
mysql> DESCRIBE C13H6;
+------------+---------------+------+-----+---------+-------+
| Field      | Type          | Null | Key | Default | Extra |
+------------+---------------+------+-----+---------+-------+
| SNo        | int(11)       | NO   |     | NULL    |       |
| X1         | double(8,5)   | YES  |     | NULL    |       |
| Y1         | double(8,5)   | YES  |     | NULL    |       |
| Z1         | double(8,5)   | YES  |     | NULL    |       |
| ATOM_NAME1 | varchar(2)    | YES  |     | NULL    |       |
| X2         | double(8,5)   | YES  |     | NULL    |       |
| Y2         | double(8,5)   | YES  |     | NULL    |       |
| Z2         | double(8,5)   | YES  |     | NULL    |       |
| ATOM_NAME2 | varchar(2)    | YES  |     | NULL    |       |
| X3         | double(8,5)   | YES  |     | NULL    |       |
| Y3         | double(8,5)   | YES  |     | NULL    |       |
| Z3         | double(8,5)   | YES  |     | NULL    |       |
| ATOM_NAME3 | varchar(2)    | YES  |     | NULL    |       |
| LARGEST    | double(8,5)   | YES  |     | NULL    |       |
| TOL1       | double(8,5)   | YES  |     | NULL    |       |
| TOL2       | double(8,5)   | YES  |     | NULL    |       |
| ANGLE1     | double(8,5)   | YES  |     | NULL    |       |
| ANG1_LIMIT | double(8,5)   | YES  |     | NULL    |       |
| ANGLE2     | double(8,5)   | YES  |     | NULL    |       |
| ANG2_LIMIT | double(8,5)   | YES  |     | NULL    |       |
| PATH       | varchar(1000) | YES  |     | NULL    |       |
+------------+---------------+------+-----+---------+-------+
21 rows in set (0.00 sec)
```

Figure 3.9: Getting information about the table

3.3. Proposed Technique

The problem of feature matching is to select a subset of points in a specific spatiotemporal domain and identify its features such that variation in the coordinates of the selected subset can be studied. Although some feature matching algorithms have come up in recent years to identify pharmacophore based search, yet the optimal tool has remained elusive.

In this dissertation work, molecules are obtained from any of the protein databases and a new numbering scheme is applied to renumber the atoms so that the proposed technique is independent of the numbering scheme already provided in the PDB file. The numbering technique is based on the number of connections an atom has. The atoms with the maximum number of connections (without the terminal atoms) are considered as the root node for numbering. A tree is formed based on the connection list. On account of how many atoms it traverses to complete the molecule, determines which atom to be finally considered for re-numbering.

Subsequently, a unique path is generated which traces the path of the entire molecule. It is required to trace the path of the entire molecule to remove the redundancy which can occur in the later stage while the unique key is generated. This unique path is used to obtain the key of the molecule. Triangles are then generated from the coordinates of each point of the molecule. The points are nothing but the coordinates of the atoms of the molecule. Thereafter, the atom keys, the largest side, its tolerance limits, the adjacent angles and their tolerance limits are calculated for each of the triangles generated. This is done by using Geometric hashing. All these information are stored into the database with the table name being the unique key of the molecule obtained in the earlier stage.

For the feature matching phase, any two molecules are selected from the database and these features (largest side, adjacent angles) are compared. If there is a complete match, a score of 1 is awarded for every feature. If the value lies in the tolerance range, a penalty is set and deducted from the score. If the value is out of the tolerance range, then a score of 0 is awarded.

3.4. Methodology

Program 1: Performing Coarse-Graining on Chemical Structures

In Coarse-Graining, the molecular structures are divided into sub-structures according to the connectivity between the atoms of the molecule. After the formation of various sub-structures of the molecule, the radius and the overall electronegativity of the respective sub-structures are calculated and stored.

Program 2: Using Depth First Search for finding out the path of the molecular structure

The molecular structures are represented as undirected labelled graphs with the atoms representing the vertices of the graph and the bonds representing the edges. The connection between the atoms is found out by finding out the distances between each of the atoms. If the distances lie within its atomic radius, which means a connection between those atoms exists. Thereafter, a graph is constructed using this information about the connections. Subsequently, the DFS algorithm is applied to find out the path through the entire molecular structure.

Program 3: Uniquely number the molecular structures on the basis of its inter-atomic connections

It is carried out in various steps:

Step 1:

Find out the terminal atoms. Terminal Atoms are those atoms which have only one atom connected to it.

Step 2:

Remove the terminal atoms from all the atom connection list. This is done to ease out the process. Now the connection list contains those atoms which have more than one connections to other atoms.

Step 3:

Extract the atom names from the connection list. Select those atoms which have the maximum number of connections and which have the least value lexicographically.

Step 4:

Run the Recursive function for each of those selected atoms.

Step 5:

After the Recursive function, the selected atoms will contain a list of the path which it follows to cover the entire molecule. This path also contains the terminal atoms. The atom containing the least number of atoms in the path is selected for renumbering. If there are more than one such atoms, it means that the molecule is symmetric. Hence, randomly any one atom can be selected for renumbering as it will not have any effect on the molecule as such.

Step 6:

The atoms of the molecule are renumbered on the basis of the order of the path. The connection list and the coordinate list are also updated accordingly.

Recursive()

1) Maintain 3 lists and 4 HashMaps.
    a) visited list: contains the atoms which have been visited
    b) finished list: contains the atoms which have finished processing
    c) unfinished list: contains the atoms which have been visited but are not processed
    d) connector1 HashMap: contains the tree of the current atom when it arrives for the first time.
    e) connector2 HashMap: contains the tree of the current atom when it is already present in connector1.
    f) connector3 HashMap: contains the tree of the current atom when it is already present in connector1 and connector2.
    g) connector4 HashMap: contains the tree of the current atom when it is already present in connector1, connector2 and connector3.
2) Construct a tree for each of the atom encountered as the current atom of the molecule. The tree structure is stored in the HashMaps where the key becomes the current atom and the value becomes the entire path.
3) After updating the entire HashMaps, the atom names are extracted. Those atoms which have terminals in the connection list are added, and an entire string is prepared which is the path of the molecule.
4) This process is repeated for all selected atoms mentioned in Step 3.

Let us take an example of a small molecule called Ethane. It has eight atoms in total: Two Carbon atoms and Six Hydrogen atoms.

Figure 3.2: The tree formation of Ethane. (a) When 1 is taken as the root node. (b) When 2 is taken as root node

In the example above (Ethane), 1 and 2 are Carbon atoms. The Terminal atoms (H) are removed from the connection list. 1 is connected to 3, 4, and 5. 2 is connected to 6, 7, and 8. Therefore, while storing the path, it will be stored as: 1, 3, 4, 5, 2, 6, 7, 8 when 1 is taken as root node and 2, 6, 7, 8, 1, 3, 4, 5 when 2 is taken as root node. This includes the terminal atoms. Now, as Ethane is a symmetric molecule, any one of the above two paths can be chosen.

If 1 is taken as the root node, the new numbering scheme would be as follows:

Table 3.1: Renumbering of Ethane when 1 is the root node

Old Numbering	1	**2**	**3**	**4**	**5**	6	7	8
New Numbering	1	**5**	**2**	**3**	**4**	6	7	8

If 2 is taken as root node, the new numbering scheme would be:

Table 3.2: Renumbering of Ethane when 2 is the root node

Old Numbering	1	2	3	4	5	6	7	8
New Numbering	**5**	**1**	**6**	**7**	**8**	**2**	**3**	**4**

Program 4: Finding out the Unique Key of the molecule

Uniquely number the molecular structures and find out a unique key for each molecule using DFS. Here, firstly the numbering scheme mentioned in the PDB file is read and analysed. A new numbering scheme is generated based on the connections of the atoms in the molecule. This is done so that the unique key is independent of any information mentioned in the PDB file because different database sites have different numberings of the same molecule. Hence, to take the provided numbering values would limit the proposed algorithm to a particular database only. After generating the numbering scheme, the unique key is generated by tracing the molecule and applying three sub-functions recursively.

The lists and MultiMaps maintained are as follows:

a) pn list-processed nodes
b) nodes_visited list-the nodes which are already in processed nodes.
c) Nodes_not_visited list-those nodes which have not been processed.
d) Terminal list: nodes containing just one connection
e) Reverse list: nodes which have terminal nodes and are visited again.
f) Neighbour MultiMap: containing the connection list of each and every atom of the molecule.

The flow-charts to all the three functions are mentioned in Appendix-A.

1) *LookUp()*
    a) This function analyses the connectivity of the given atom and places it in the processed node path if it has not already been visited. The reading of the neighbour list is done in increasing order of the numbering.
    b) The neighbours which are not being visited are placed in the nodes_not_visited list, and the nodes which are being visited are placed in the nodes_visited list.
    c) If there is only one neighbour of the given atom which has been already visited then that neighbour is visited again and is placed in the reverse list. The atom is placed in the terminal list.

d) If all the neighbours are present in the nodes_visited list, then call the *RightShift* function.
e) If all the nodes have been visited then exit.

2) *RightShift*
   a) This function reads the neighbour list of the given node in decreasing order of numbering.
   b) The node to be considered for the next atom in the processed node should not be an immediate node (the node from where it is coming) and it should not be present in the nodes_visited list or in the reverse list or in the terminal list. Add that node to the nodes_visited list.
   c) Add the neighbours which are not being visited to the nodes_not_visited list. That node should also be not present in the nodes_visited list.
   d) If there is only one neighbour of the given atom which has been already visited then that neighbour is visited again and is placed in the reverse list. The atom is placed in the terminal list.
   e) Repeat the process until and unless all the neighbours are present in the (reverse list U terminal list) →*LifeLine* function or there is a node in the neighbour list which have not been visited→*LookUp* function or the nodes_not_visited list is empty→exit.

3) *LifeLine*
   a) Check if there is an unvisited node. If yes call the *LookUp* function. If no, add the node which is the intersection of the nodes_not_visited list and the neighbour list.
   b) If there is only one neighbour of the given atom which has been already visited then that neighbour is visited again and is placed in the reverse list. The atom is placed in the terminal list.
   c) If there is no intersection between the nodes_not_visited list and the neighbour list, then go back one step and write that node in the processed node list.
   d) Repeat the process until the entire nodes_not_visited list is empty, or all the nodes are present in the nodes_visited list. Then the process stops.

Let us take an example of Ethane. The path generated is as follows:

Table 3.3: Table describing the operation of the Unique Key Generation Algorithm

Atom No.	Nodes Not Visited	Nodes Visited	Neighbours	Immediate Node	Reverse	Terminal Atoms	Comments
1	-	-	2,3,4,5	-	-	-	LookUp Function called.
2	1,~~2~~,3,4,5	1	1	1	-	2	Only 1 neighbour(1), visit and place it in reverse list
1	~~1~~,3,4,5	1,2	2,3,4,5	2	1	2	
3	~~3~~,4,5	1,2	1	1	1	2,3	Only 1 neighbour(1), visit and place it in reverse list
1	4,5	1,2,3	2,3,4,5	3	1	2,3	
4	4,5	1,2,3	1	1	1	2,3,4	Only 1 neighbour(1), visit and place it in reverse list
1	5	1,2,3,4	2,3,4,5	4	1	2,3,4	
5	~~5~~	1,2,3,4	1,6,7,8	1	1	2,3,4	

Atom No.	Nodes Not Visited	Nodes Visited	Neighbours	Immediate Node	Reverse	Terminal Atoms	Comments
6	1,~~6~~,7,8	1,2,3,4,5	5	5	1	2,3,4,6	Only 1 neighbour(5), visit and place it in reverse list
5	1,7,8	1,2,3,4,5,6	1,6,7,8	6	1,5	2,3,4,6	
7	1,~~7~~,8	1,2,3,4,5,6	5	5	1,5	2,3,4,6,7	Only 1 neighbour(5), visit and place it in reverse list
5	1,8	1,2,3,4,5,6,7	1,6,7,8	7	1,5	2,3,4,6,7	
8	1,~~8~~	1,2,3,4,5,6,7	5	5	1,5	2,3,4,6,7,8	Only 1 neighbour(5), visit and place it in reverse list
5	1	1,2,3,4,5,6,7,8	1,6,7,8	8	1,5	2,3,4,6,7,8	Process stops. All nodes visited

Program 5: Geometric Hashing

Find the number of possible triangles from the number of points submitted as an input (co-ordinates of the molecule in concern), finding the longest side of each triangle and both the adjacent angles of the longest side using Geometric Hashing. Also, the tolerance limits of each of the sides as well as the adjacent angles are found out. After that, store this information into the MySQL database.

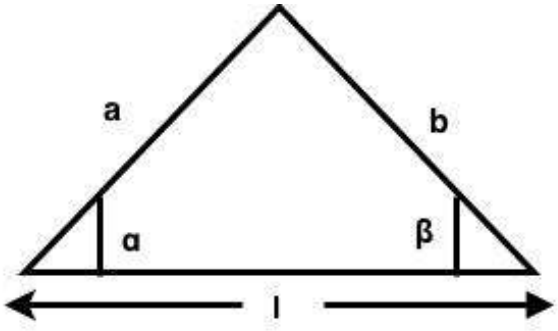


Figure 3.3: A triangle showing the largest side and adjacent angles

l→largest side of the triangle

a & b → adjacent sides of the longest side

The equations for finding out the angles is:

$$\alpha = \cos\frac{a^2+l^2-b^2}{2*l*a} \quad 3.1$$

$$\beta = \cos\frac{b^2+l^2-a^2}{2*l*a} \quad 3.2$$

Where α,β → adjacent angles of the largest side

The equation for finding out the tolerance limits of each side is as follows:

$$T_n = 0.15 * R_n \quad 3.3$$

Where T→Tolerance limit

n→1,2,3

R→ Atomic Radius of the atom

For finding out the tolerance limits of the adjacent angles, firstly a perpendicular is drawn to the largest side from the opposite angle of the largest side.

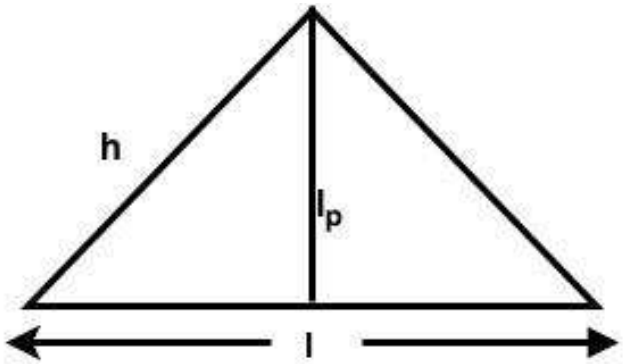


Figure 3.4: Triangle showing the perpendicular drawn to the largest side

l_p → length of the perpendicular

h → hypotenuse of the triangle

l_p and h are calculated using the Pythagoras Theorem. Using l_p now, the maximum and minimum angles of each of the adjacent angles can be calculated.

When maximum tolerance limit is to be calculated l_p is increased by adding T_3 to it.

When minimum tolerance limit is to be calculated l_p is decreased by subtracting T_3 from it.

$$new\_angles = \sin^{-1} \frac{new\_l_p}{h} \qquad 3.4$$

Where new_angles→ maximum or minimum range of the angle

new_l_p→ l_p + T_3 (for maximum angle)

l_p - T_3 (for minimum angle)

The tolerance limits for the adjacent angles can, now, be easily calculated by subtracting the maximum angle from the actual angle found out earlier.

Program 6: Feature matching of two given molecules.

In this program, two molecules are selected from the given database and the triangles having same atom sequences are matched. The criteria for the matching algorithm are the largest sides and the adjacent angles. Accordingly, a score is calculated, and a percentage of how similar the 1st molecule is to the 2nd molecule is taken out. If the values are exactly same, then a score of 1 is awarded. If the values lie outside the tolerance limits, a score of 0 is awarded.

If there is a slight change in the values (that is, the values lie within the tolerance limit), a penalty is deducted from the score. The formula for taking out the penalty is as follows:

$$P = 1 - \frac{large - small}{large\_tol} \quad 3.5$$

Where P=Penalty

large → the atom having the larger value

small → the atom having the smaller value

large_tol → the tolerance limit of the larger value

CHAPTER 4
RESULTS AND DISCUSSIONS

Program 1:

The output of the Coarse-Graining program is as follows:

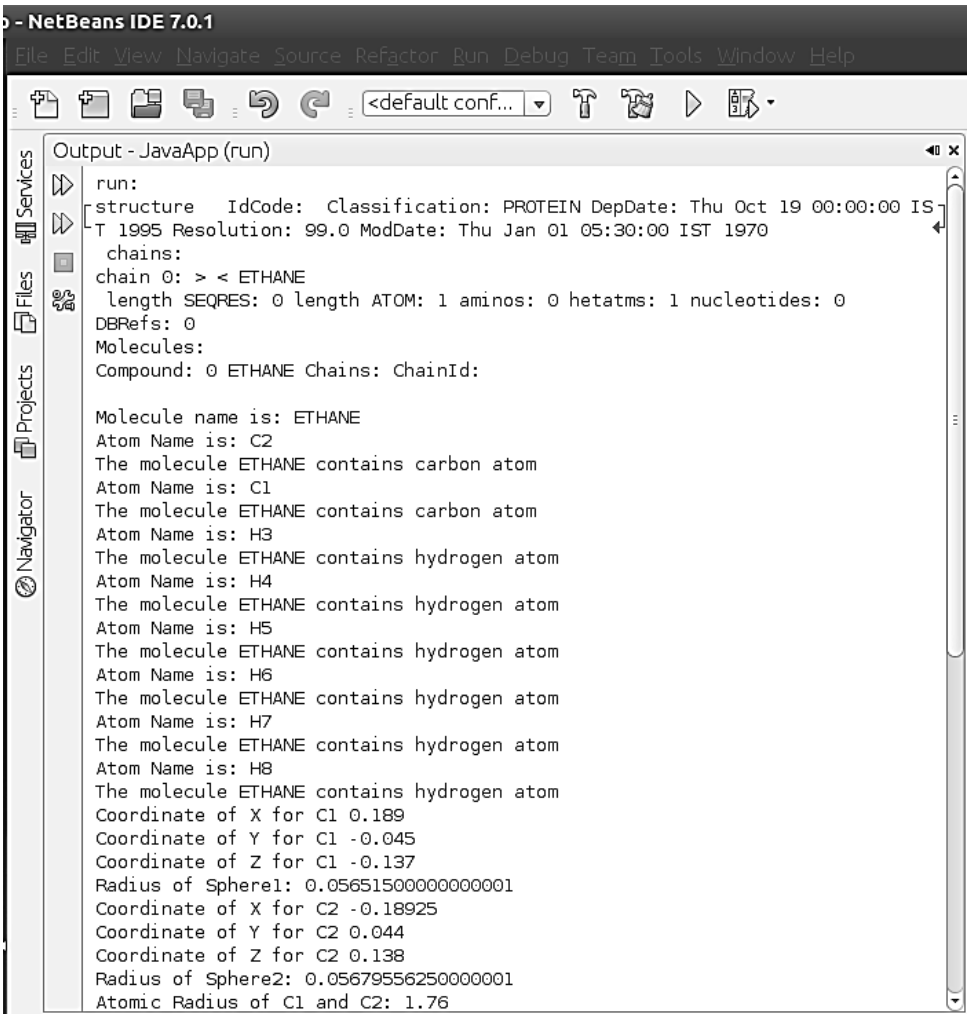


Figure 4.5: Output of Coarse-Graining Algorithm

From the first screenshot, it can be understood that molecules are taken from any of the PDB sites. Thereafter, the atoms are categorized on the basis of their connections. The radius and electro-negativity of these categories are then determined. In the example above of Ethane, the Carbon connections are found out. Here, there are two Carbon atoms. Hence, this molecule is divided into 2 sub-type. Each of the sub-molecule contains a Carbon atom and three Hydrogen atoms. The radii of both the sub-molecules are calculated by finding out the maximum distance between the Carbon atom and Hydrogen atom. The electro-

negativity is then calculated by finding out the electro-negativity of each of the atoms in the sub-molecules and averaging them.

Program 2:

The output of DFS algorithm is:

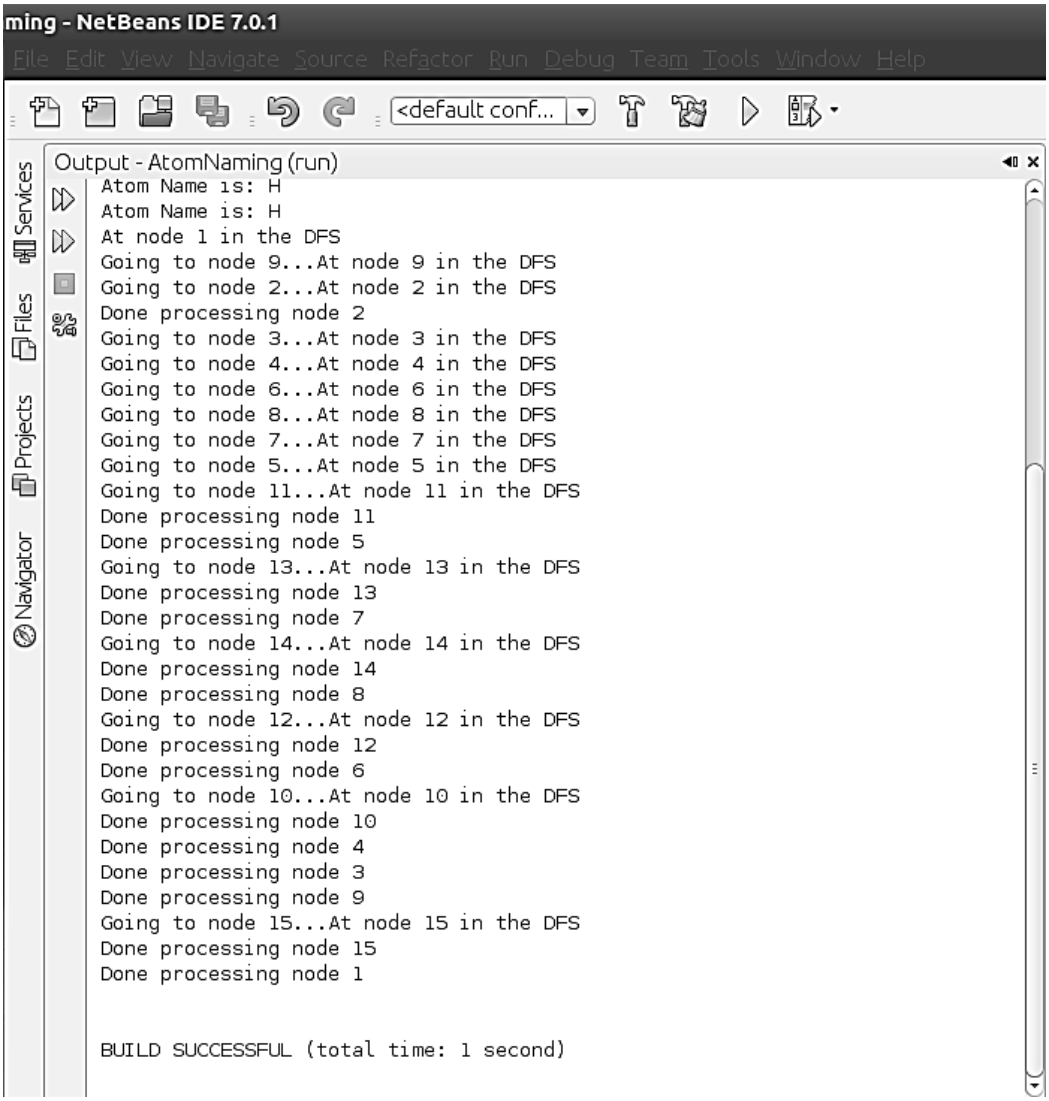


Figure 4.6: Output of the DFS Algorithm

This is an example of Benzoic Acid. There are fifteen atoms in Benzoic Acid. The path to the entire molecule is traced based on the Depth-First Search Algorithm.

Program 3:

The output of the Numbering program is as follows:

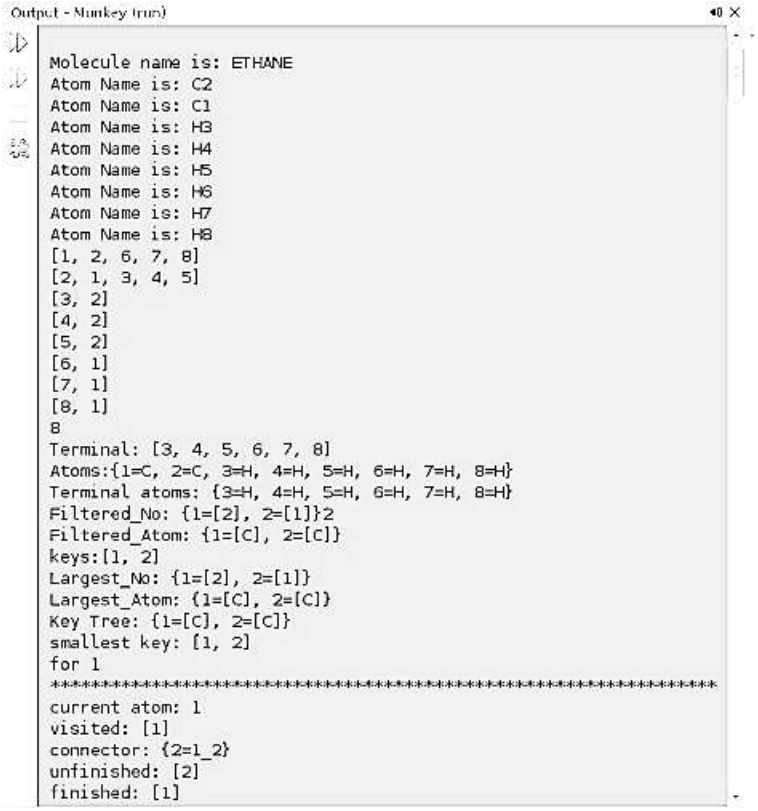


Figure 4.7: Screenshot of the Numbering Algorithm on Ethane (1)

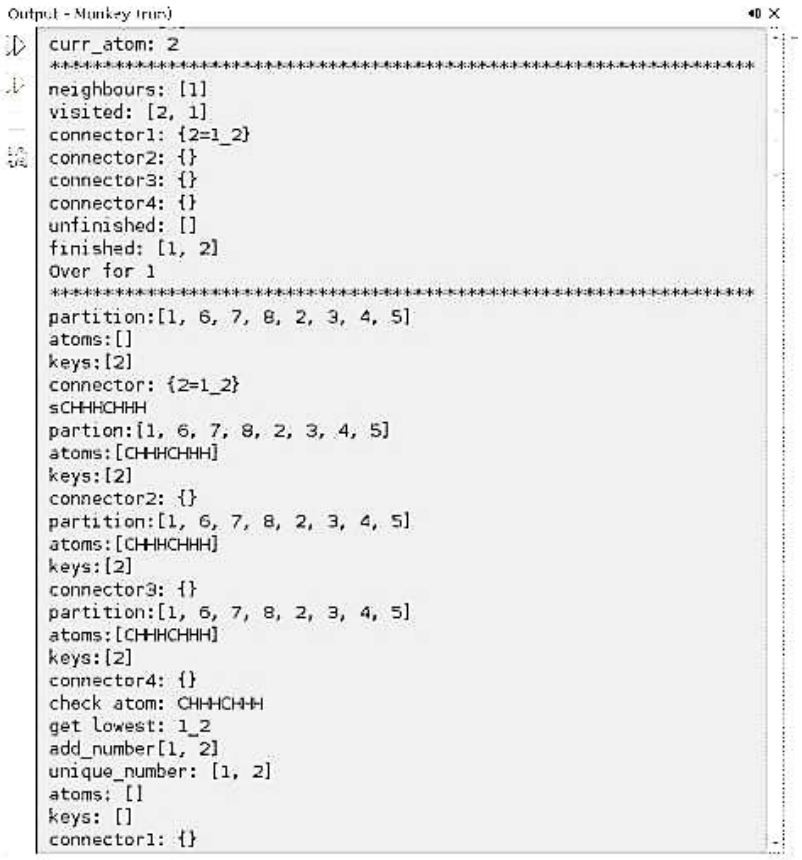


Figure 4.8: Screenshot of the Numbering Algorithm (2)

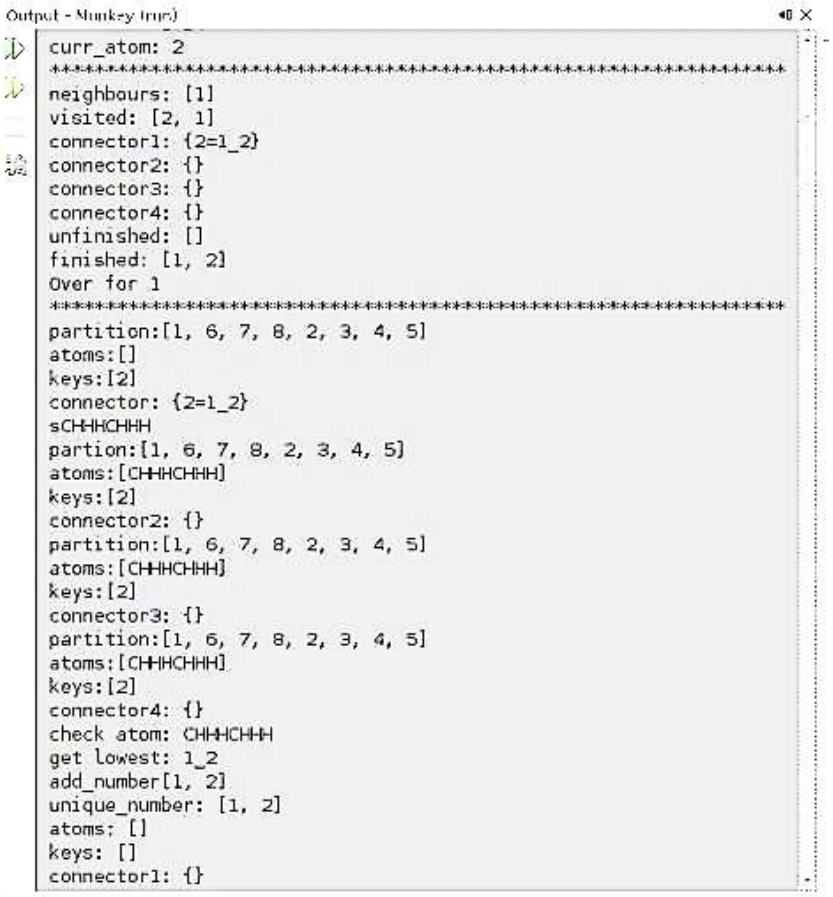


Figure 4.9: Screenshot of the Numbering Algorithm (3)

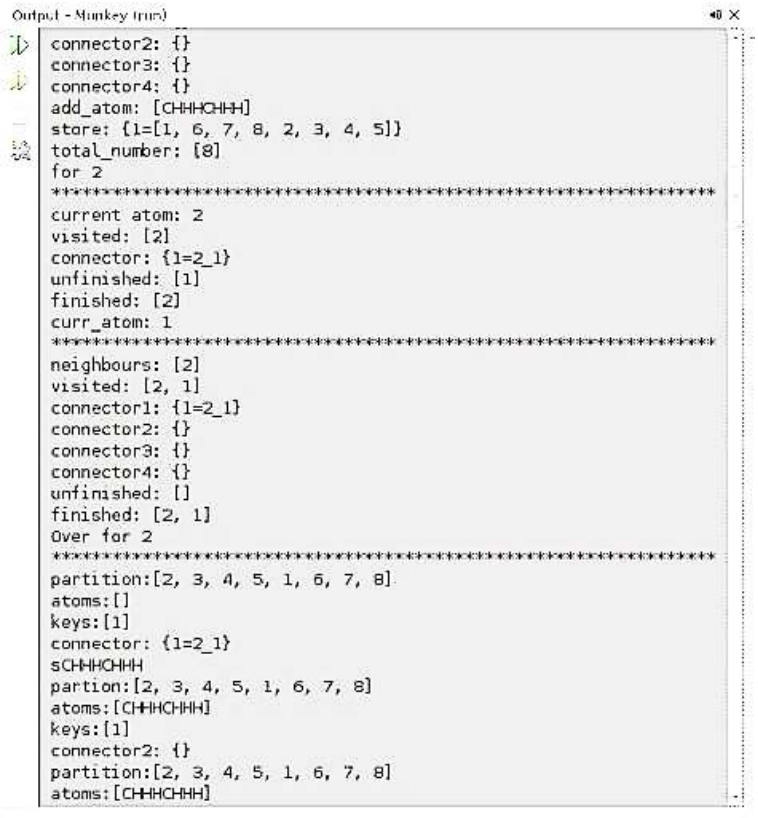


Figure 4.10: Screenshot of the Numbering Algorithm (4)

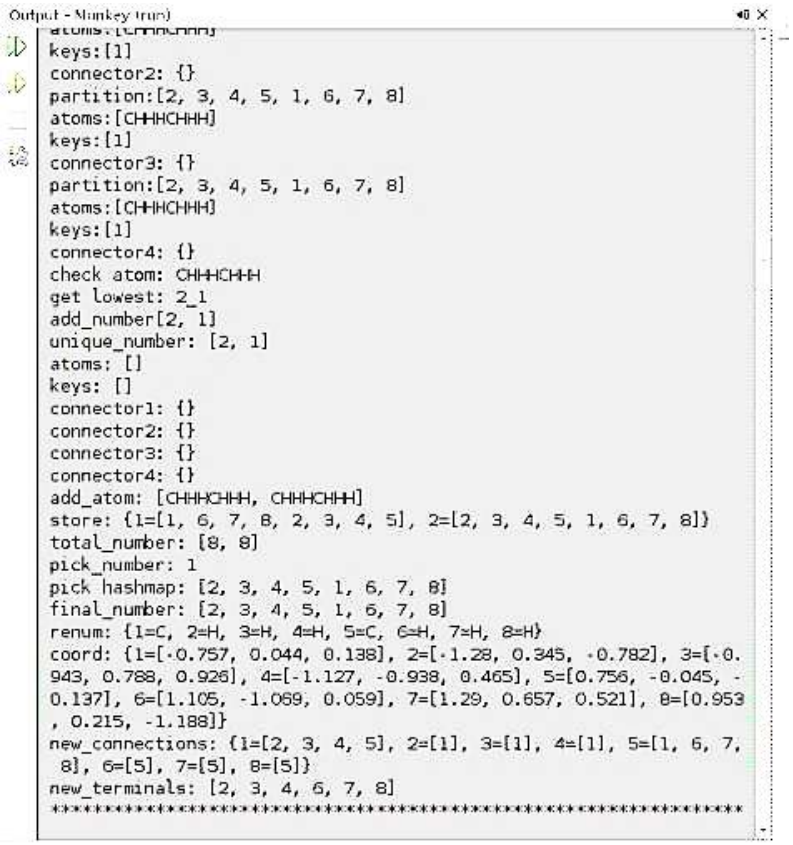


Figure 4.11: Screenshot of the Numbering Algorithm (5)

In these screenshots, it is understood that Ethane molecule is taken, and the new numbering scheme is applied which is explained in the Methodology Section.

Program 4:

The output of the Unique Key Algorithm is as follows:

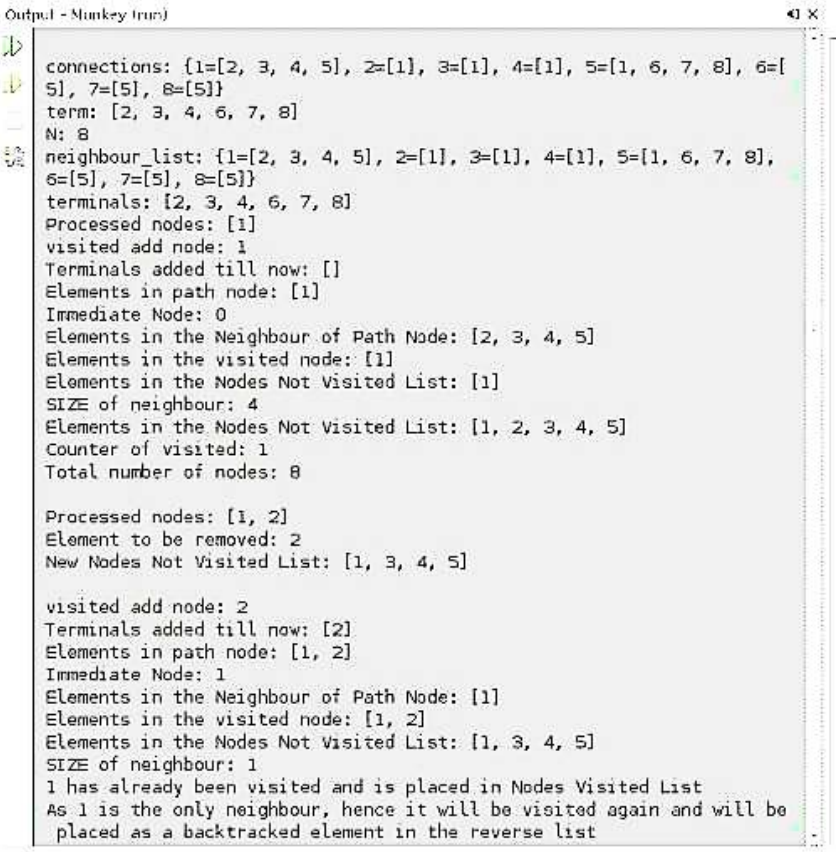


Figure 4.12: Screenshot of the Unique Key Algorithm (1)

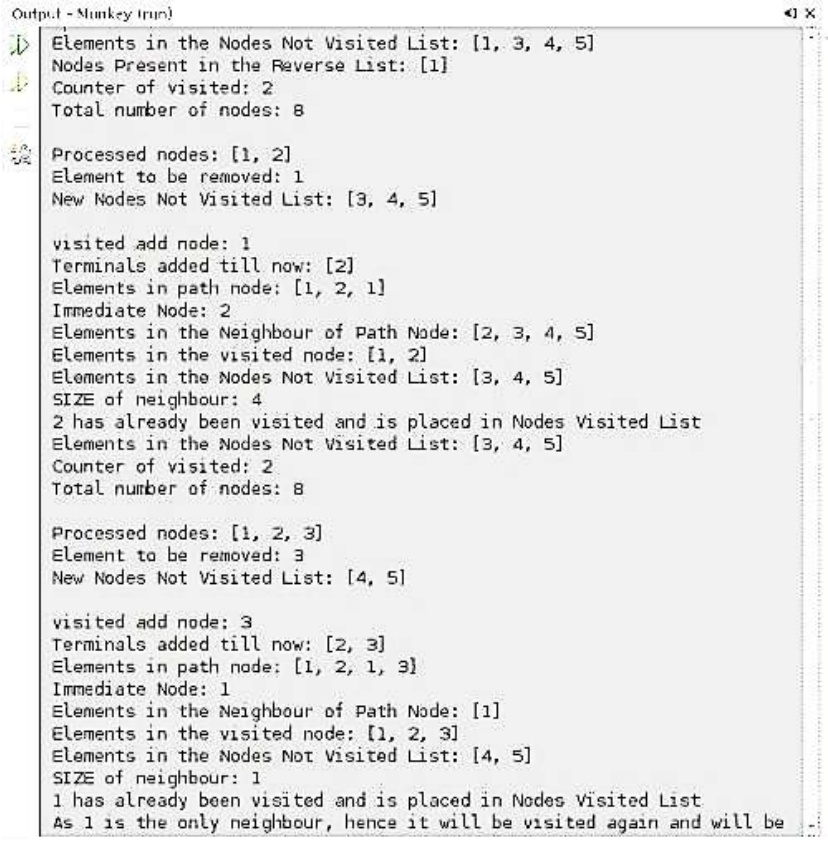


Figure 4.13: Screenshot of the Unique Key Algorithm (2)

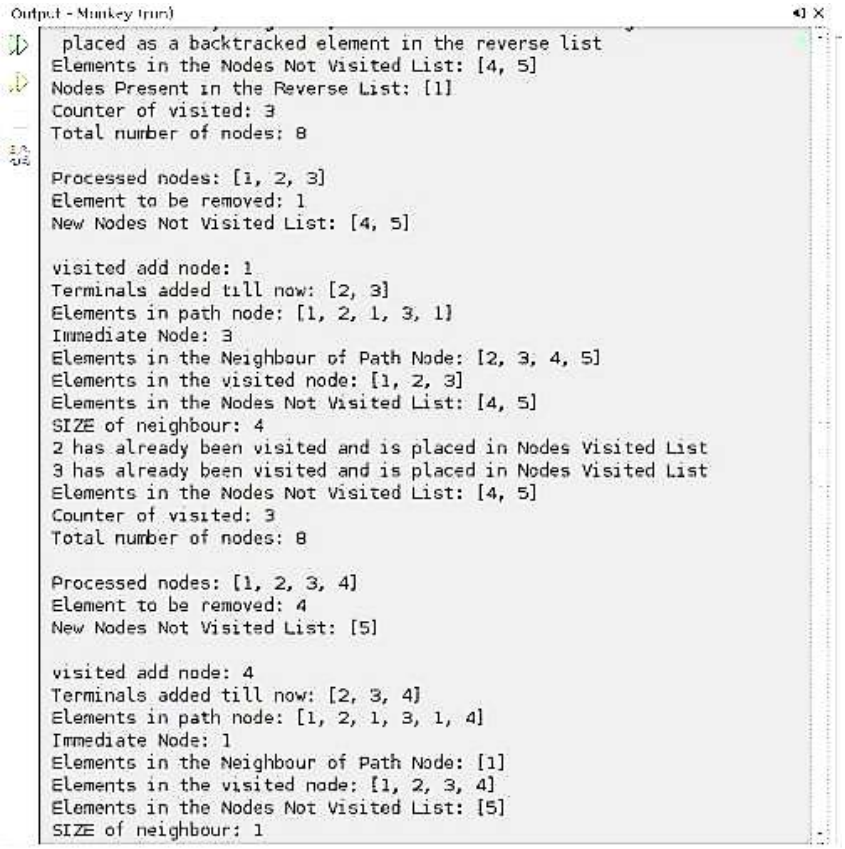


Figure 4.14: Screenshot of the Unique Key Algorithm (3)

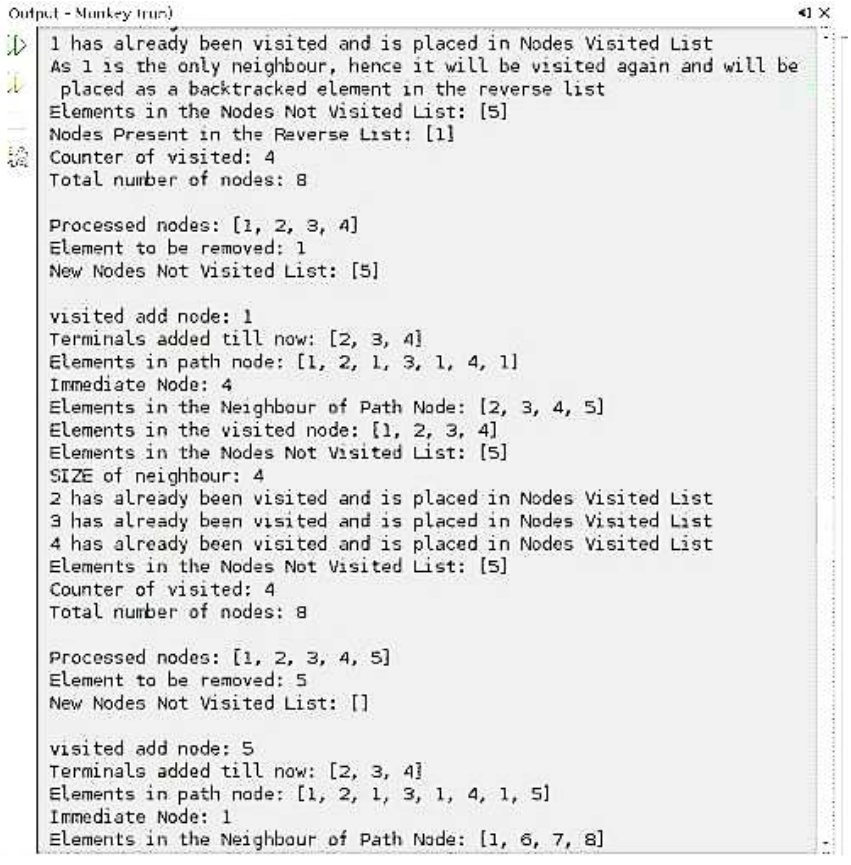


Figure 4.15: Screenshot of the Unique Key Algorithm (4)

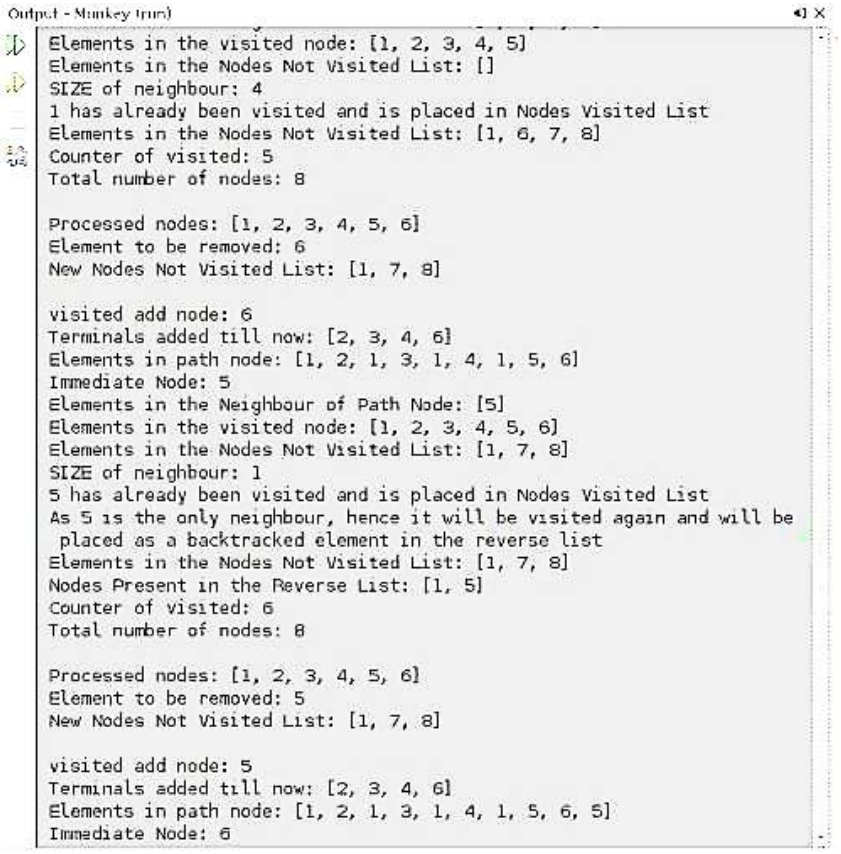


Figure 4.16: Screenshot of the Unique Key Algorithm (5)

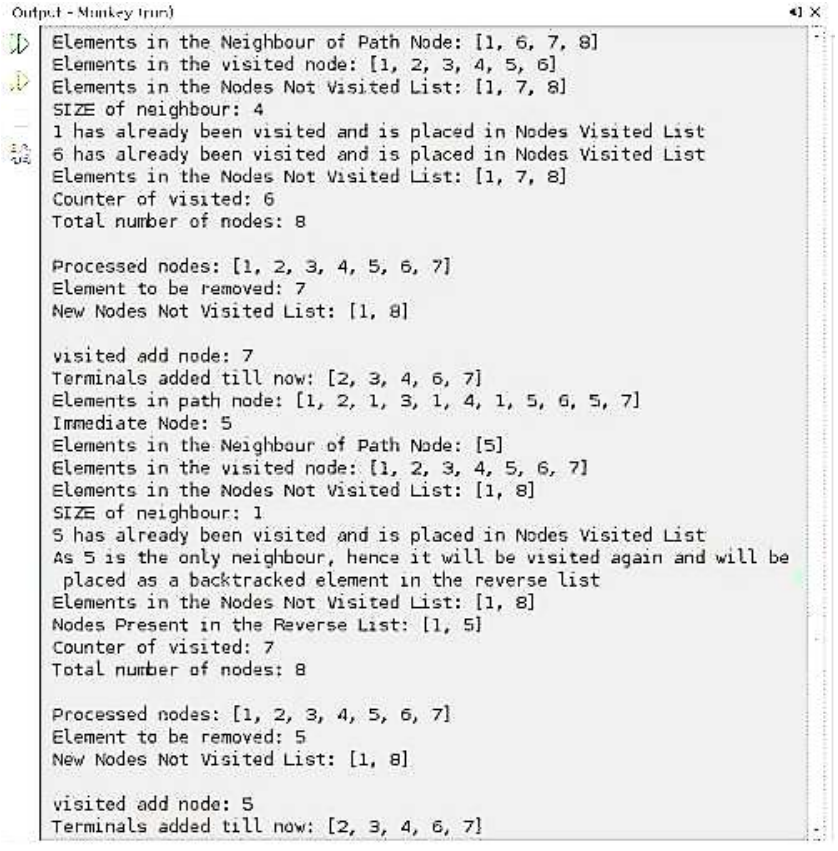


Figure 4.17: Screenshot of the Unique Key Algorithm (6)

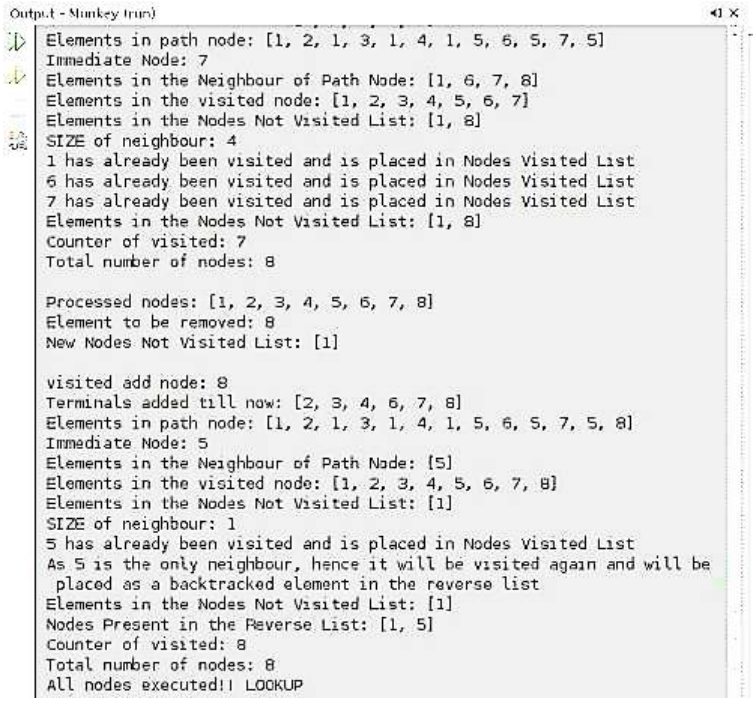


Figure 4.18: Screenshot of the Unique Key Algorithm (7)

In the above screenshots, it is seen that the unique path of Ethane is traced by using the three main functions: LookUp, RightShift and LifeLine. The algorithm stops when all the atoms have been visited.

Program 6:

The output of the Geometric Hashing Algorithm is as follows:

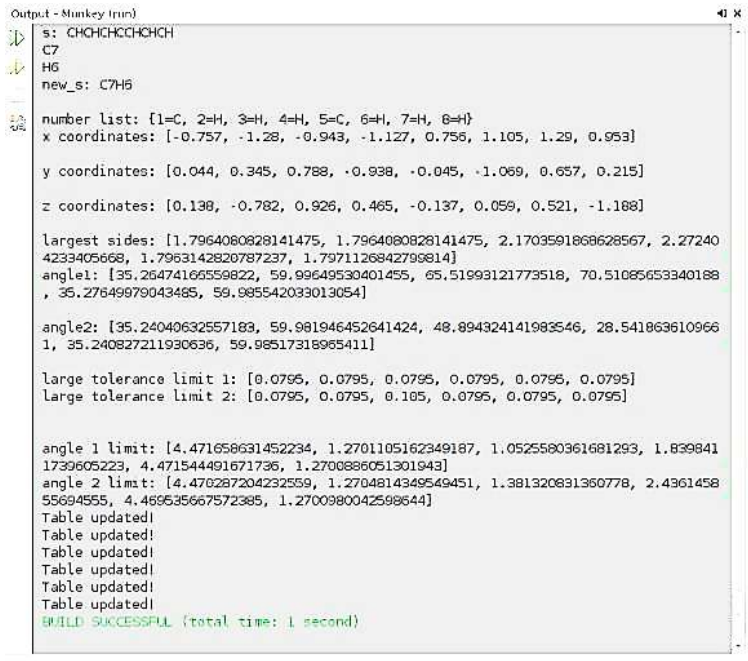


Figure 4.19: Output of the Geometric Hashing Algorithm

In this case, an example of Ethane is taken. The unique key of Ethane, calculated using the Unique Key Algorithm, is CHCHCHCCHCHCH. With the atoms of each molecule, a triangle is generated. The table name for Ethane would be C7H6. The largest side, its tolerance limits, the adjacent angles and their respective tolerance limits are calculated, and all these information are stored in the database having the table name as C7H6. If there already exists a table called C7H6 then the unique path of both the tables are checked (the existing table in the database and the new one). If there is a match, it means that Ethane is already there in the database. The updation in this case will not occur. If both the unique keys are different, it means that both the molecules are different, and the new molecule will be inserted into the database having the table name C7H6_A. Likewise, if another molecule is inserted with the same table name, then it will be stored as C7H6_B and so on. Therefore, data redundancy is removed.

Program 6:

The output for the Feature Matching Algorithm are as follows:

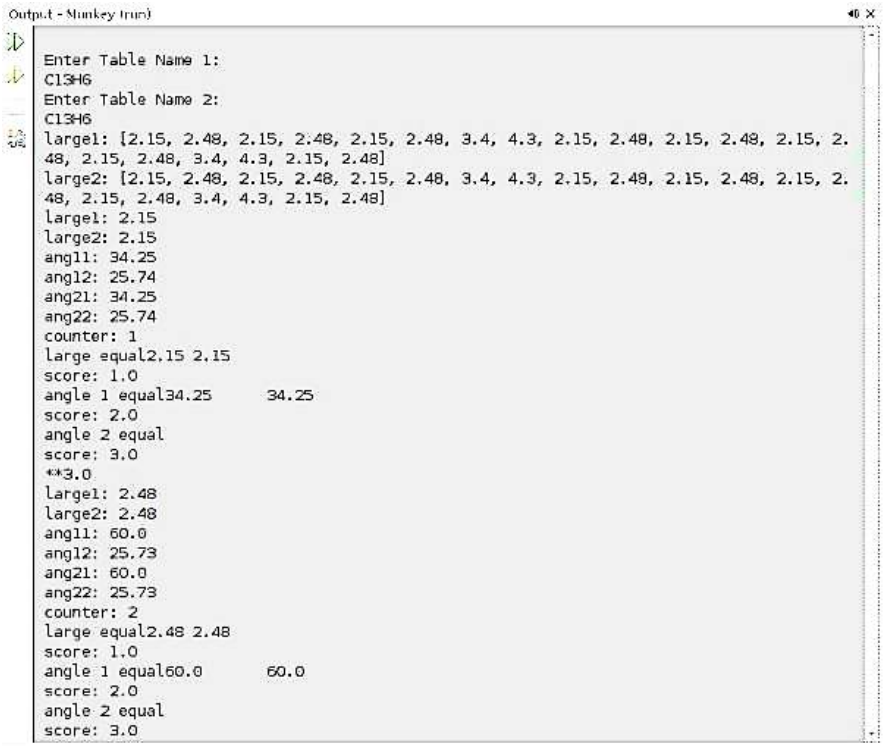


Figure 4.20: Output of the Feature Matching Algorithm when there is an exact match (the molecule is same) (1)

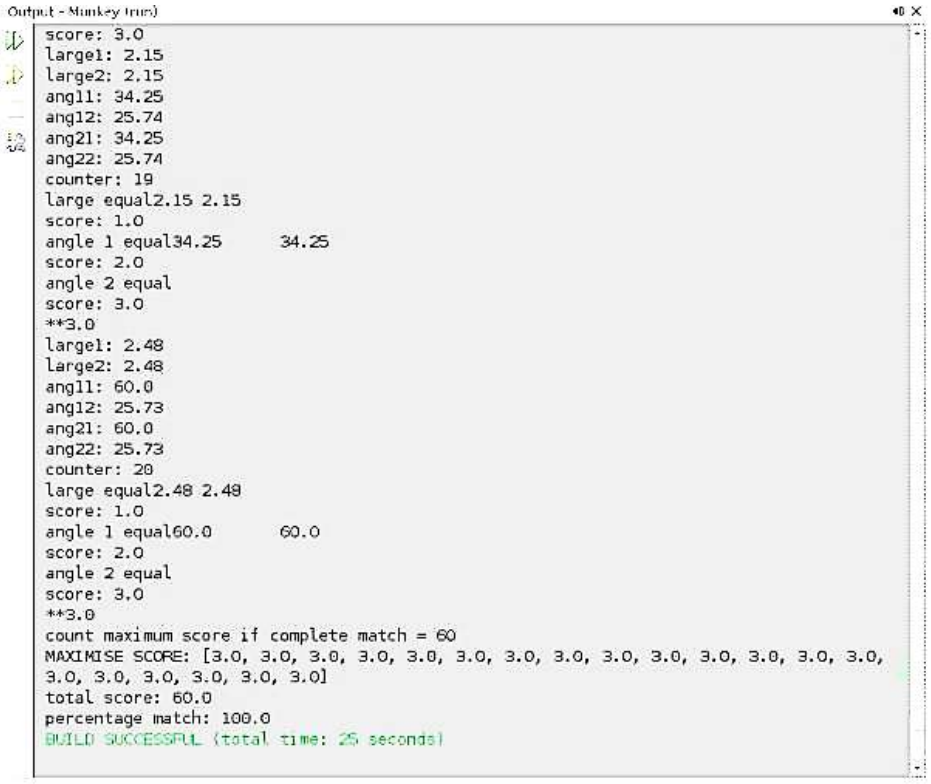


Figure 4.21: Output of the Feature Matching Algorithm when there is an exact match (the molecule is same) (2)

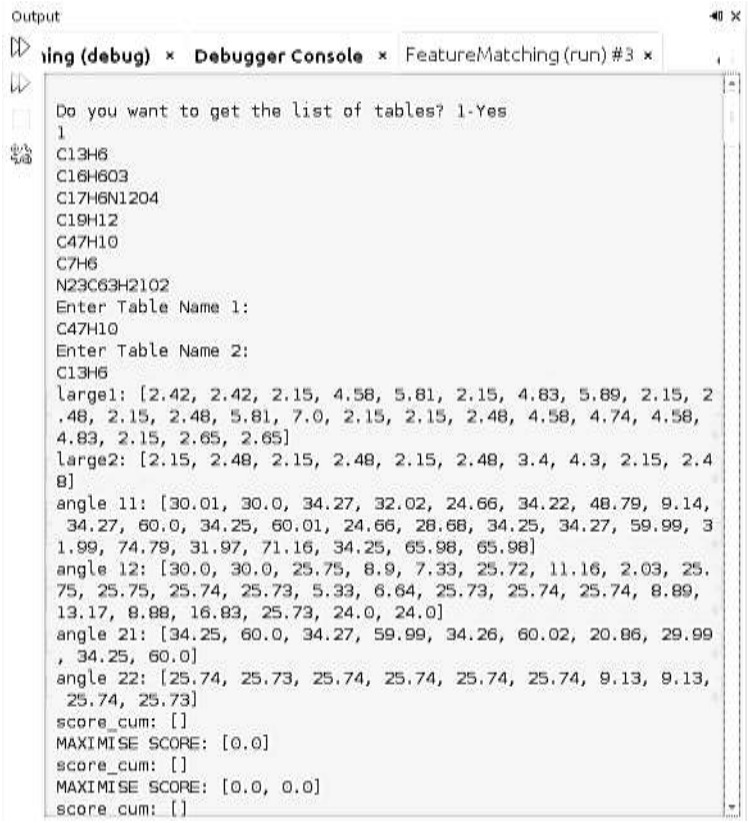


Figure 4.22: Output of the Feature Matching Algorithm when there are two different molecules (benzene and pyrene in this case) (1)

Figure 4.23: Output of the Feature Matching Algorithm when there are two different molecules (benzene and pyrene in this case) (2)

In the feature matching phase, two tables are selected from the database whose similarity needs to be checked. The atom sequence is matched in the first stage. Thereafter, the largest sides and the adjacent angles are analysed. Accordingly, a score is calculated, and a percentage of similarity is generated. The True Positive Rate and the False Positive Rates are then calculated.

$$TP = 1$$

$$TN = 1$$

$$FP = 0$$

$$FN = 0$$

$$TPR = \frac{TP}{TP+FN} = \frac{TP}{P} = 1$$

$$FPR = \frac{FP}{FP+TN} = \frac{FP}{N} = 0$$

Hence, Accuracy, $ACC = \frac{TP+TN}{P+N} = 1$

Thus, the objective of the dissertation work gets fulfilled. The main advantage of this kind of feature matching is that it overcomes the conformational barrier.

CHAPTER 5
CONCLUSION

In this dissertation work, a number of molecules are taken from the protein databases and a new numbering scheme is applied. Subsequently, a unique path is generated which traces the path of the entire molecule. This unique path is used to obtain the key of the molecule. Triangles are then generated from the coordinates of each point of the molecule. The points are nothing but the coordinates of the atoms of the molecule. Thereafter, the largest side, its tolerance limits, the adjacent angles and their tolerance limits are calculated for each of the triangles generated. All these information are stored into the database with the table name being the unique key of the molecule. For the feature matching phase, any two molecules are selected from the database and these features (largest side, adjacent angles) are compared. If there is a complete match, a score of 1 is awarded. If the value lies in the tolerance range, a penalty is set and deducted from the score. If the value is out of the tolerance range also, then a score of 0 is awarded. This algorithm based on feature matching is precise and accurate. The main advantage of this algorithm is that it overcomes the conformational barrier which was previously not achieved by any of the algorithms. The problem of conformational space enormity is critical for shape matching algorithms. The algorithms, presently in use, attempt to overcome this by sampling conformational space. However, in the current approach, the need for conformational sampling is altogether abolished without compromising the matching efficiency. This improves the “similar compounds detection” speed.

FUTURE WORK

This algorithm does not distinguish between single bonds and multiple bonds. In future, there can be an approach to differentiate between types of bonds. Secondly, this algorithm is based on atom-centric matching. A better approach would be to convert groups of atoms into a bigger feature and then perform feature-based matching. Thirdly, this algorithm works for smaller molecules. Modifications in this algorithm can perform for larger molecules with more than thousand atoms as in the case of proteins, ligands, etc.

REFERENCES

Drie, J. H. Van. (2010). History of 3D pharmacophore searching: commercial, academic and open-source tools. *Drug Discovery Today. Technologies*, **7**(4), 203–270.

Drie, J. H. Van (1997). Strategies for the determination of pharmacophoric 3D database queries. *Journal of Computer-Aided Molecular Design*, **11**(1), 39–52.

Ebalunode, J. O., Ouyang, Z., Liang, J., & Zheng, W. (2008). Novel approach to structure-based pharmacophore search using computational geometry and shape matching techniques. *Journal of Chemical Information and Modeling*, **48**(4), 889–901.

Grossman, R., Hamelberg, D., Kasturi, P., & Liu, B. (2008). Experimental studies of the universal chemical key (UCK) algorithm on the NCI database of chemical compounds. Computational Systems Bioinformatics. CSB2003. Proceedings of the 2003 IEEE Bioinformatics Conference. CSB2003, 244–250.

Hawkins, P. C. D., Skillman, a G., Warren, G. L., Ellingson, B. a, & Stahl, M. T. (2010). Conformer generation with OMEGA: algorithm and validation using high-quality structures from the Protein Databank and Cambridge Structural Database. *Journal of Chemical Information and Modeling*, **50**(4), 572–84.

Jenkins, J. L., Glick, M., & Davies, J. W. (2004). A 3D similarity method for scaffold hopping from known drugs or natural ligands to new chemotypes. *Journal of Medicinal Chemistry*, **47**(25), 6144–59.

Jones, G., Willett, P., & Glen, R. C. (1995). A genetic algorithm for flexible molecular overlay and pharmacophore elucidation. *Journal of Computer-Aided Molecular Design*, **9**(6), 532–49.

Kevin, L. J. (2008). Undergraduate Research Opportunity Research (UROP) Project Report Linear Representation of Graphs. Computing, 1–25.

Kier, M. (2007). Molecular Design, 271–279.

Landrum, G. (2012). RDKit Documentation.

Loew WEee Kheng. (2007). *Feature Detection and Matching*, 1–38.

Keller, M.G. (1999). *Matching Algorithms and Feature Match Quality Measures for Model-Based Object Recognition*.

Mestres, J., Rohrer, D. C., & Maggiora, G. M. (1997). A molecular field-based similarity approach to pharmacophoric pattern recognition. *Journal of Molecular Graphics & Modelling*, **15**(2), 114–121.

O'Boyle, N. M. (2012). Towards a Universal SMILES representation - A standard method to generate canonical SMILES based on the InChI. *Journal of Cheminformatics*, **4**(1), 22-37.

Spjuth, O. (2007). Bioclipse.

Srinivasan, A., Page, D., Camacho, R., & King, R. (2006). Quantitative pharmacophore models with inductive logic programming, **64**(1-3), 65–90.

Steindl, T., & Langer, T. (2004). Influenza virus neuraminidase inhibitors: generation and comparison of structure-based and common feature pharmacophore hypotheses and their application in virtual screening. *Journal of Chemical Information and Computer Sciences*, **44**(5), 1849–1856.

Tarjan, R. (1971). Depth-first search and linear graph algorithms. 12th Annual Symposium on Switching and Automata Theory (swat 1971), **1**(2), 146–160.

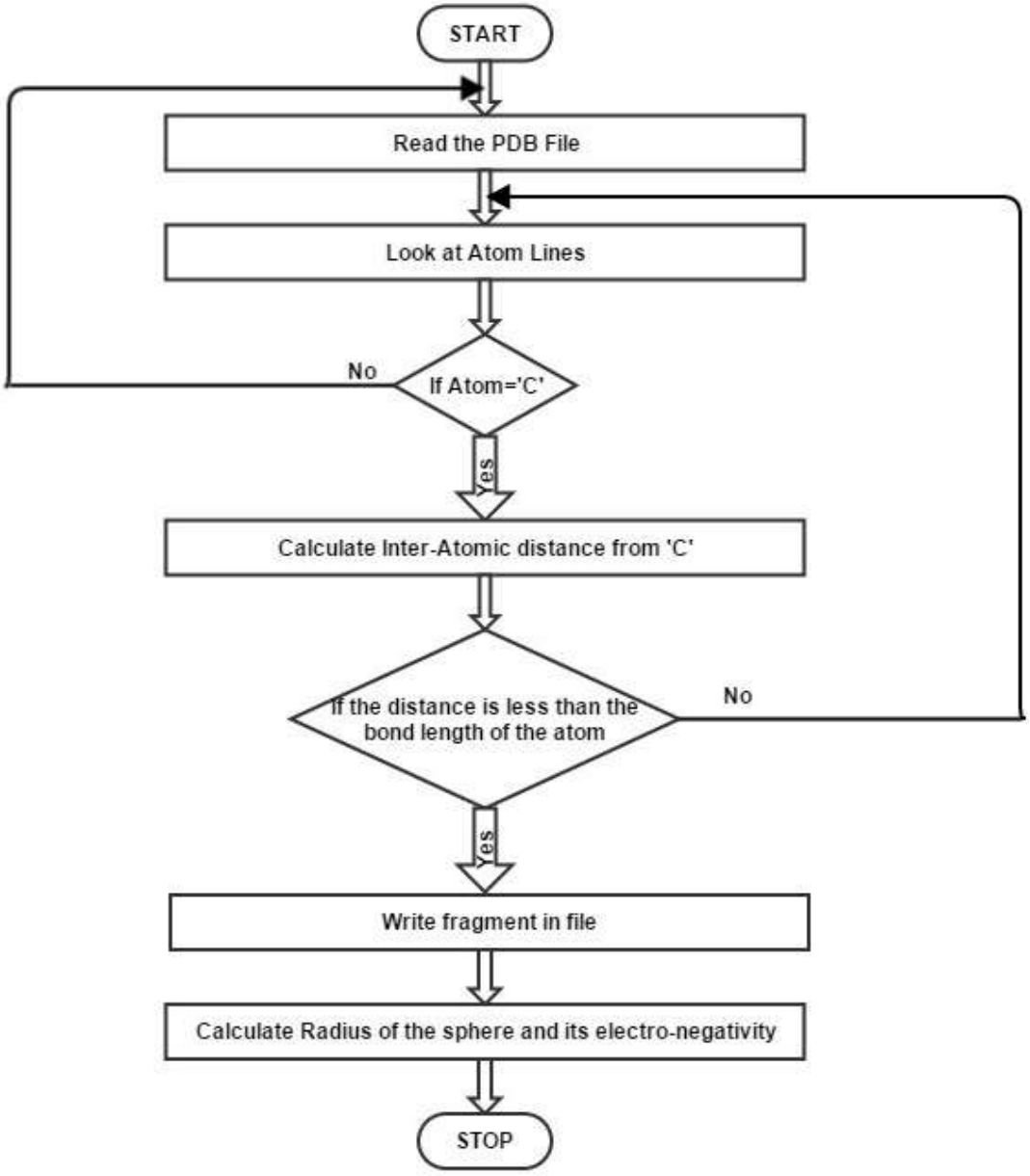
Weininger, D., Weininger, A., & Weininger, J. L. (1989). SMILES: Algorithm for Generation of Unique SMILES Notation. Journal of Chemical Information and Computer Sciences, **29**(2), 97–101.

APPENDIX A

In this section, the various flowcharts of the programs mentioned in the Methodology Section of Chapter 3 are provided.

Program 1:

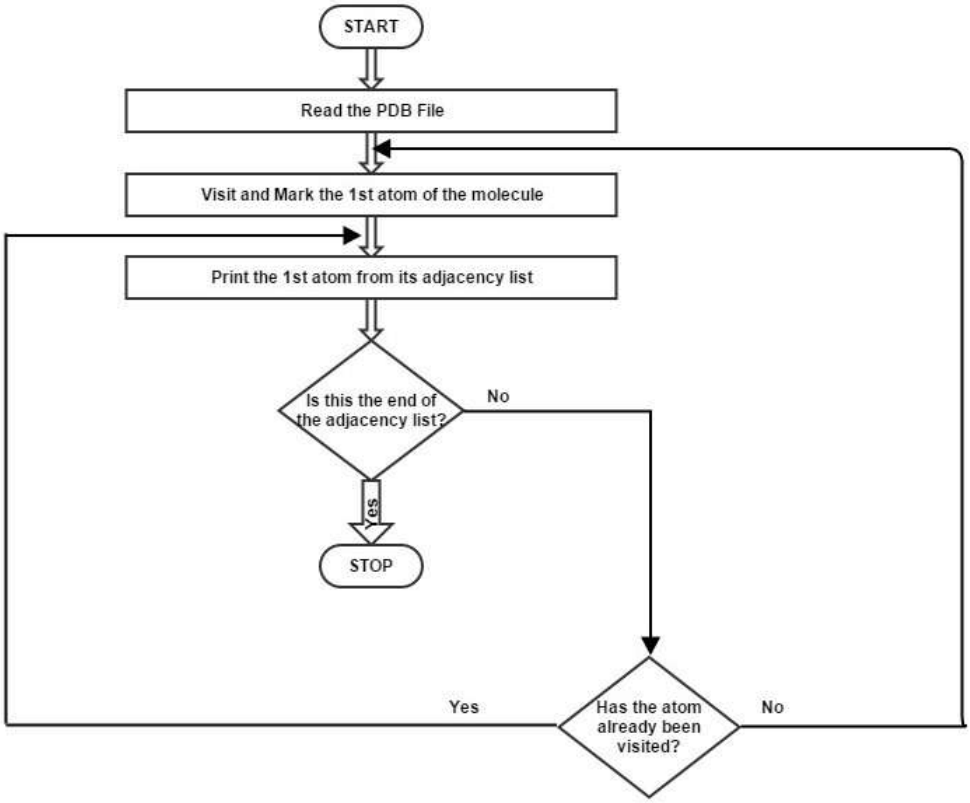
The flowchart for the Coarse-Graining Algorithm is as follows:



Coarse-Graining Flowchart

Program 2:

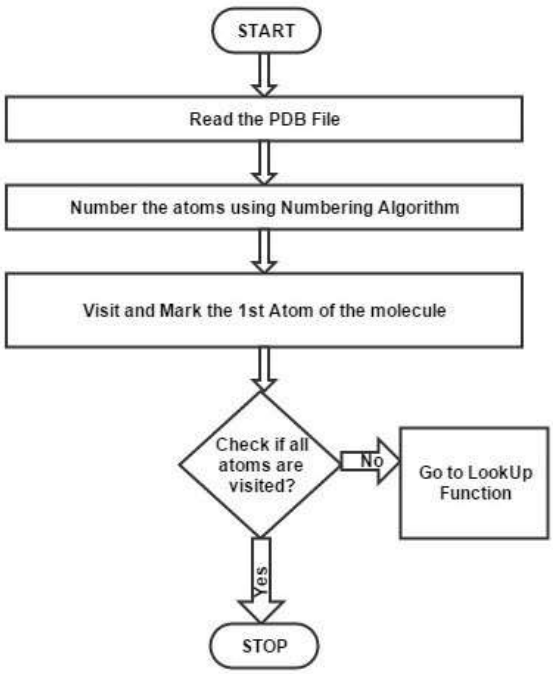
The flowchart for the Depth-First Search Algorithm is as follows:



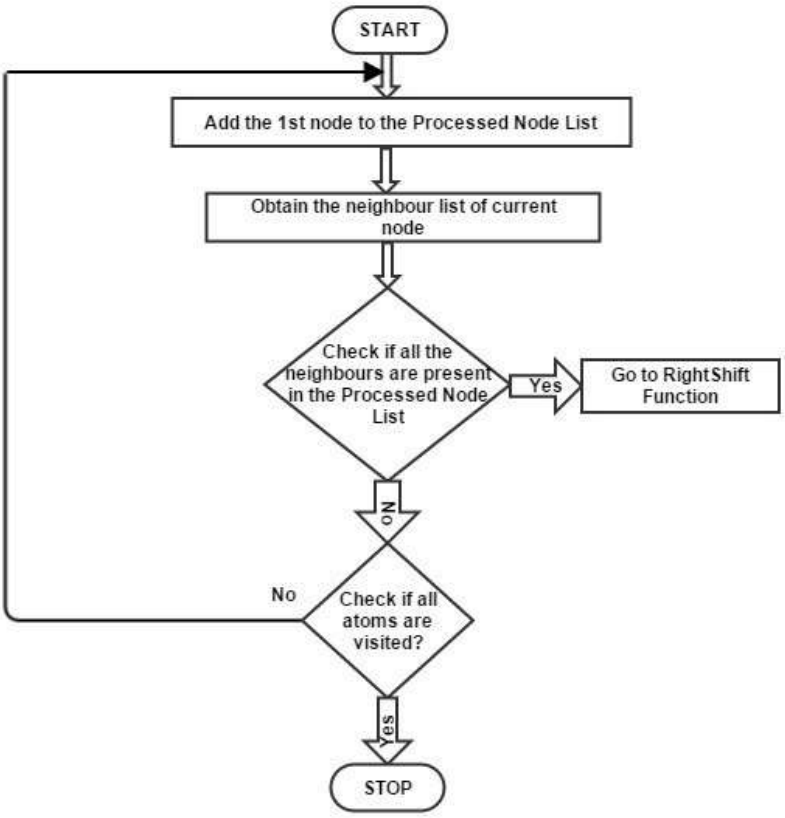
Depth First Search Flowchart

Program 3:

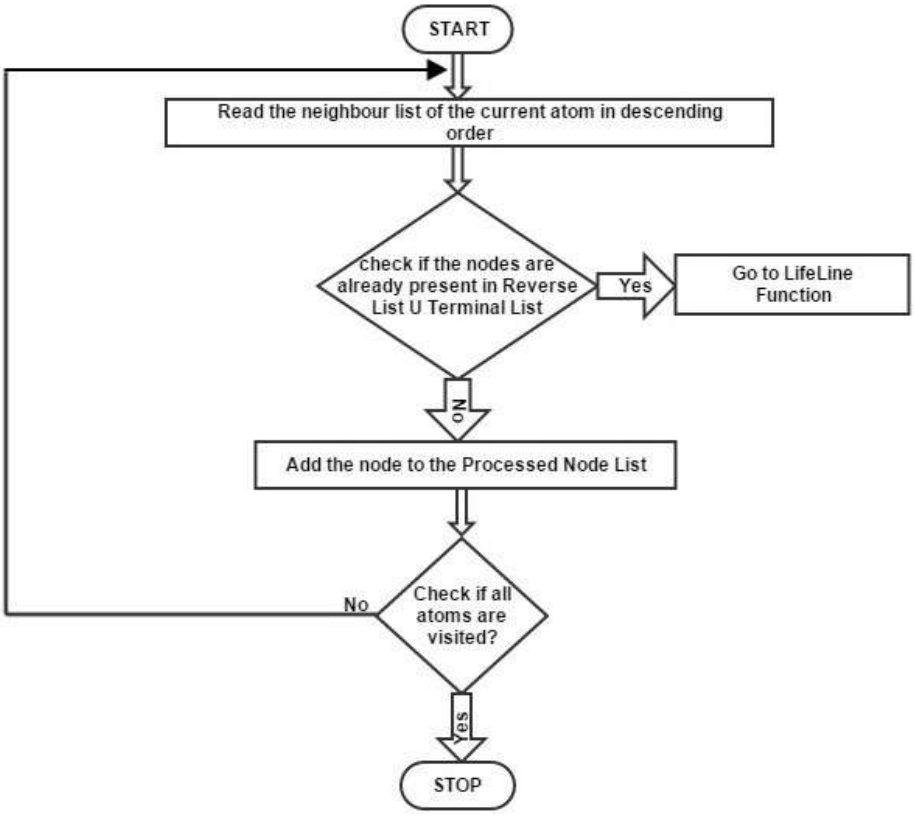
The flowcharts for the Unique Key Generation are as follows:



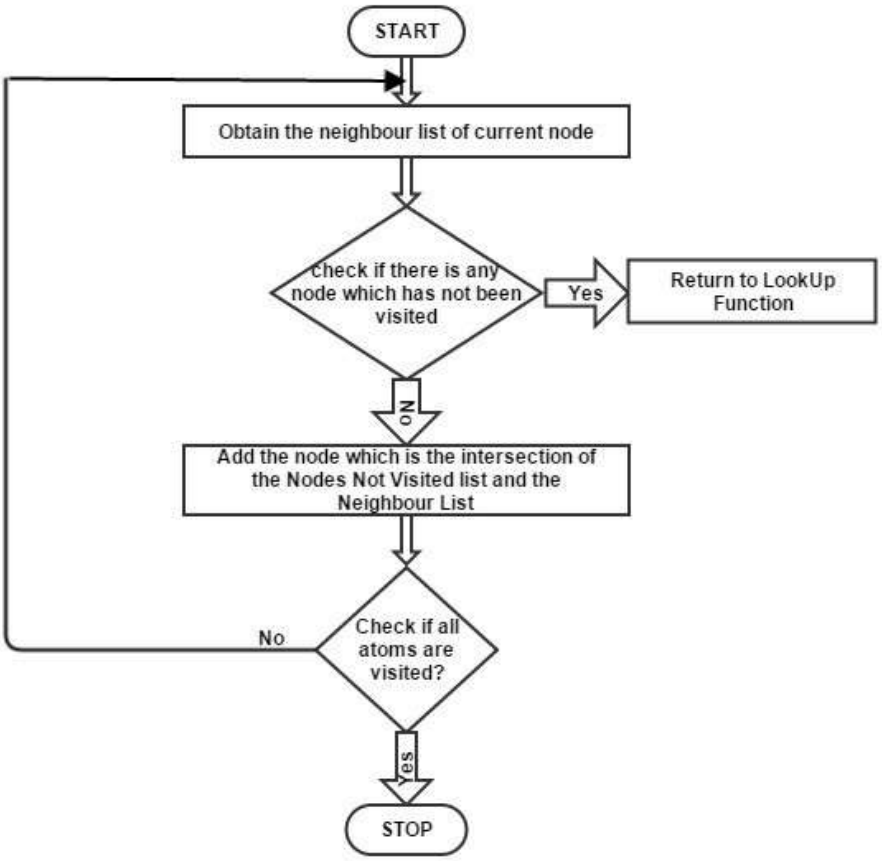
Unique Key Generation Algorithm



Flowchart for LookUp Function



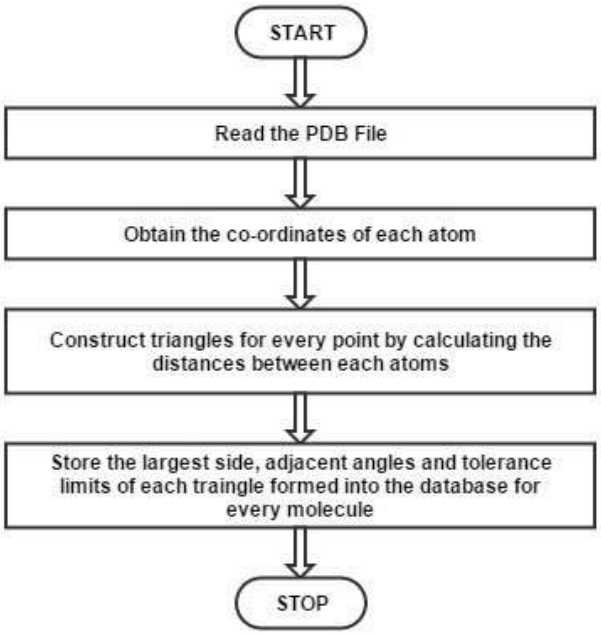
Flowchart for RightShift Function



Flowchart for LifeLine Function

Program 4:

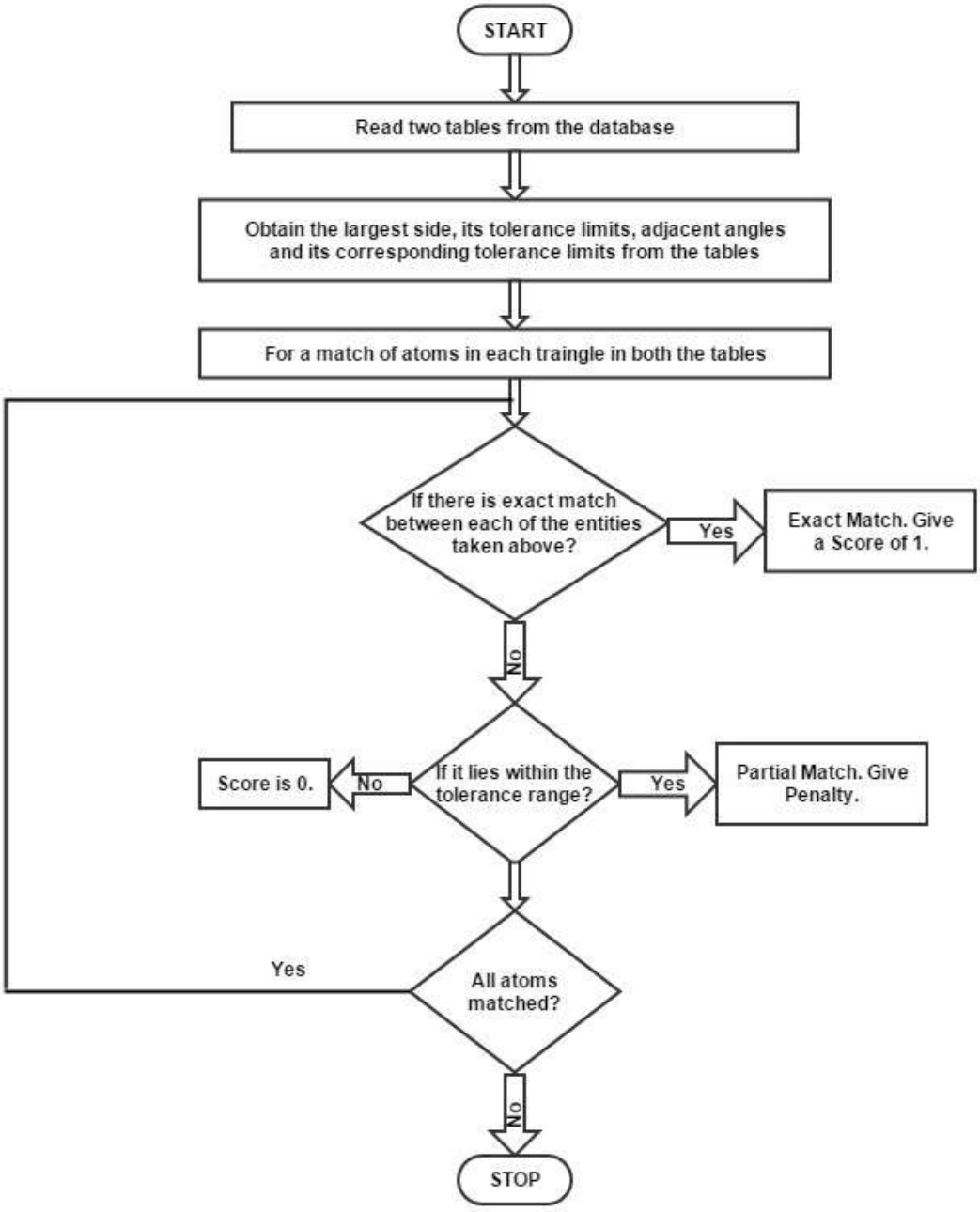
The flowchart for the Geometric Hashing Algorithm is as follows:



Geometric Hashing Flowchart

Program 5:

The flowchart for the Feature Matching Algorithm is as follows:



Feature Matching Flowchart