Chemical Structure Searching in Chemical Compound Graph Databases

Aye Nwe Thaing
University of Computer Studies, Yangon
ayenwethaing@gmail.com

Abstract

In chemical informatics, the structure of chemical compound can be represented as graph with atoms as nodes and bonds as edges. Exact structural graph matching (automorphism), exact substructure graph matching (subgraph isomorphism) and graph similarity searching are important research areas of chemical informatics. In this paper, we develop the system for efficient exact graph matching and graph similarity searching using the edge code. The structural information of chemical graph is generated as the edge code. To improve the precision of exact graph matching, we use our novel edge code index structure to find the automorphic graph in the database. For similarity searching, we utilize the fuzzy query similarity graph matching based on the edge code to improve the computational speed. Finally, we conduct an extensive set of experiments on chemical compound database to demonstrate the efficiency of our approach for efficient exact graph matching and similarity searching.

1. Introduction

Chemistry is the science of matter, especially its chemical reactions but also its composition, structure and properties[11]. Chemistry is concerned with atoms and their relative interactions with other atoms and particularly with the properties of chemical bonds. A chemical compound is a substance with a particular ratio of atoms of particular chemical elements which determines its composition, and a particular organization which determines chemical properties. For example, water is a compound containing hydrogen and oxygen in the ration of two to one, with the oxygen atom between the two hydrogen atoms. Compounds are formed and inter-converted by chemical reactions [11].

Development of chemical structure databases for use in research and development area is a well-established activity in the pharmaceutical and chemical industries. There are three categories of useful tasks in the development of chemical compound databases. These are interactive search and retrieval of structures in such databases including exact structure searching, substructure searching and similarity searching. A better solution is appropriate to offer the user that would automatically find molecules containing structures nearly match the original query, and structures or substructures exact match the query [2].

For similarity searching, instead of searching for all molecules containing a given substructure, the users search for molecules “similar” to a given target molecule. Structurally similar molecules are expected to exhibit similar properties or biological activities. For substructure searching, the users would interest in finding the database since it contains a

substructure isomorphic to the query structure. In exact structure matching, the users interest to find all molecules in the compound which are identical at some level of description.

In this paper, we develop a system to process exact graph matching and graph similarity searching efficiently. The edge code is used to powerfully match exact graph structure. Fuzzy query similarity graph searching [9][5] is used to search the similar graph and faster query processing time based on the edge code.

The rest of the paper is organized as follows. Section 2 presents the related work of the proposed system. In Section 3, we present the preliminary concepts of labeled graph, exact graph matching, exact subgraph matching, similarity graph searching and about of edit distance. Section 4 discusses the representation of chemical structures into graph data model. Section 5 discusses about our proposed system. Section 6 describes exact graph matching over chemical compound database. In Section 7, we discuss graph similarity searching over chemical compound database. In Section 8, we discuss the experimental result of our proposed system. Section 9 concludes of our paper.

2. Related Work

The notion of chemical similarity is one of the most important concepts in chemical informatics[1][10]. It plays an important role in modern approaches to predicting the properties of chemical compounds, designing chemicals with a predefined set of properties and especially, in conducting drug design studies by screening large databases containing structures of available chemicals. Recently, various graph indexing methods have been designed to capture the intrinsic similarity of graphs to do the various queries in an efficient way[9][3][7][4][6].

In [11], a novel kernel-based similarity measurement has been developed to measure similarity of graph represented chemicals. G-hash method is used to support new graph kernel function, efficient storage and fast search. The utility of G-hash achieves state-of-the-art performance for *k*-nearest neighbor classification. Moreover, the similarity measurement and the index structure are scalable to large chemical databases with smaller indexing size and faster query processing time.

A new way of indexing a large database of graphs and processing exact subgraph matching and approximate graph matching queries is proposed in [3]. Each graph in the database is represented by its graph signature. During query processing, a query graph is mapped into its signature. To improve the precision of exact subgraph matching, a new method based on the concept of line graphs is proposed. The graph edit distance over graph signatures is developed for approximate graph matching. GiS can also provide a scalable and efficient disk-based solution for indexing and querying graphs. GiS can also outperform state-of-the-art techniques.

Substructure similarity search using indexed features in graph databases has been developed in [12]. Grafil filters many graphs without performing pair-wise similarity computations by transforming the edge relaxation ratio of a query graph into the maximum allowed missing features. By examining the effect of different feature selection mechanisms, a multi-filter composition strategy is developed where each filter uses a distinct and complementary subset of the features. Grafil can also be applied to searching approximate non-consecutive sequences, trees, and other complicated structures as well.

In [8], a mechanism is proposed that can check whether two graphs are automorphic or not. Storing the graphs into large databases is a challenging task as it deals with efficient space

and time management. A proficient F-GAF algorithm is designed that efficiently detects and avoids the same graph getting stored into the database based on grid code representation of a graph. The computational time is substantially reduced compared to the canonical labeling approach used in frequent subgraph discovery algorithm.

3. Preliminaries

This section describes the formal graph definitions and notations used for this work.

Definition 1 (Labeled Graph) A labeled graph G is defined as a 4-tuple, (V,E,L_v,L_e,l) where V is the set of vertices, $E \in V \times V$ is the set of edges, L_v and L_e are the set of labels for vertices and edges and l is a labeling function assigning a label to a vertex $l: V \rightarrow L_v$ or an edge $l: E \rightarrow L_e$.

Definition 2 (Exact Graph Matching): Given a query graph Q, an exact graph matching query finds the graph in the database that is automorphic to Q.

Definition 3 (Exact Subgraph Matching): Given a query graph Q, an exact subgraph matching query finds the graphs in the database that contain a subgraph that is isomorphic to Q.

Definition 4 (Similarity Graph searching): Given a query graph Q and a distance d, similarity graph matching finds the graphs in the database whose edit distance with Q is at most d.

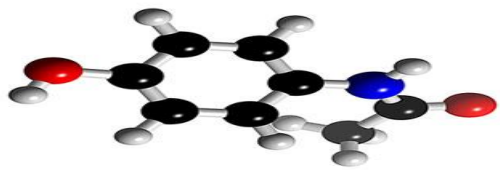
Definition 5 (Edit Distance): The edit distance d is the number of edits (insertions, deletions, and substitutions) required to transform a string (A) into another string (B). In graph similarity searching, the graphs returned from the database are within a user-specified edit distance d from a query graph.

4. Representation of Chemical Structures into graph data model

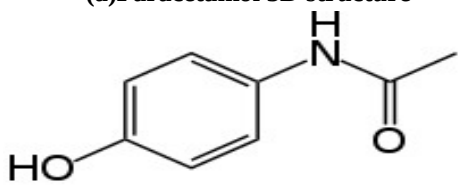
A model is a simplified approximation of reality. Scientific models are simplified but useful representations of something real (for example, 3D structures chemical compounds). However, the models are not always physical entities. Sometimes they are sets of ideas instead. But it is required to represent these scientific models to get exact structure as much as possible.

There has been a tremendous increase in our understanding of the physical world, but much of that understanding is based on extremely complicated ideas and mathematics. The application of the most sophisticated forms of these modern ideas is difficult and not very useful to those of us who are not well trained in modern physics and high-level mathematics. Therefore, scientists have developed simplified models for visualizing, explaining, and predicting physical phenomena.

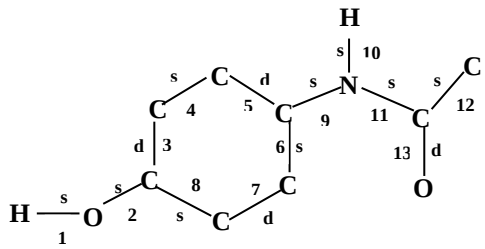
In this paper, we propose the system that combines scientific model and high-level mathematics to develop useful representation and querying of chemical compounds in graph data structure in the area of chemical informatics. Figure 1 shows the example of chemical compound paracetamol with 2D and 3D structures and graph structure of paracetamol.



(a)Paracetamol 3D structure



(b)Paracetamol 2D structure



(c)Graph structure of Paracetamol(G_{Para})

Figure 1. An example illustration of Paracetamol compound

5. Proposed System

We have developed a system that can index and query labeled, undirected graphs. The system is designed for the chemical compound graphs in the area of chemical informatics. The purposes of our proposed system are described as follows:
(1) we can verify the substances contained in the medical products that have the sufficient amount of chemical elements in definite proportion by weight
(2) we can search the chemical compounds that are nearly similar to the compounds with respect to either structural or functional qualities already defined in pharmaceutical and chemical industries.

Our key technical contributions in this work are:
(1) exact graph structure matching using edge code index structure to improve the precision of graph matching
(2) graph similarity searching based on fuzzy query graph similarity matching over edge code to improve computational speed

The architecture of our proposed system is shown in figure 2. The main components of our proposed system are edge dictionary, edge code generating engine and graph matching engine.

When the graph G enters into the graph database, the edge list of G is retrieved from the database. The edge dictionary contains the edges in the database with unique identifiers. The edge code generating engine generates the edge code which contains the nearest neighbor edges in the graph. Then the edge codes are stored in the disk storage. When a query graph Q enters into the system, the edge code of the query graph is generated and inputs to the edge code matching engine. The matching engine can process two types of queries: exact structure graph query and similarity search query. According to the user defined query mode, the matching engine processes the query and the results are displayed to the user. Table 1describes the notations used in the algorithms for our proposed system.

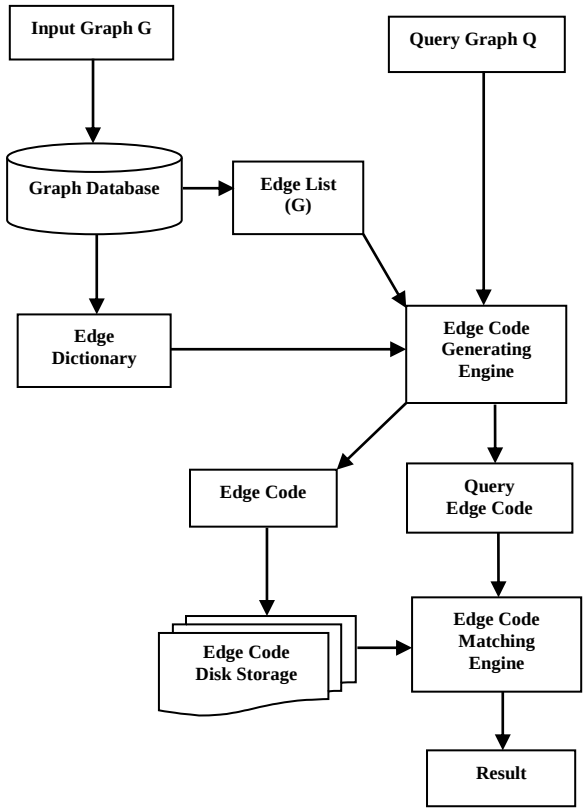


Figure 2.Architecture of the proposed system

Table 1. Notations used in the system

Notation	Definition
GDB	Graph database
G_i	Graph in the GDB
$EC(G_i)$	Edge code of G_i
$D_e(G_i)$	Distinct edge in G_i
e_{adj}	Adjacent edge of an edge e
sim_{min}	Minimum similarity value for G_i
Sim_{edit}	Similarity between G_i and G_q

5.1. Edge Dictionary

The edge dictionary contains the distinct edges in the graph database. When a graph introduces to the database, the graph may contain the same edges and needs to find distinct edges(D_e) in this graph. The edge dictionary contains the unique edges to represent the same edges in the database. Each edge in the database can be assigned with the global unique identifier from the edge dictionary for further graph processing. Therefore, it is efficient to retrieve the equivalent edge came along in the graph. In the edge dictionary, an edge is defined as a 3-tuple where ($l_{(u)}$,$l_{(u,v)}$,$l_{(v)}$) and $l_{(u)}$ and $l_{(v)}$ are the labels of the vertices and $l_{(u,v)}$ is the label of the edge itself. Table 2 shows constructing the edge dictionary using the graph shown in figure 1(c).

Table 2. Edge dictionary

ID	Edge
1	H,s,O
2	O,s,C
3	C,d,C
4	C,s,C
5	C,s,N
6	H,s,N
7	C,d,O

5.2. Edge Code Generating Engine

A graph is transformed into an edge code that captures the structural representation of the graph. The edge code generating engine computes adjacent edge information of each edge appeared in the graph. Every edge in the graph is assigned with global unique identifier already defined in the edge dictionary. For each edge e=($l_{(u)}$,$l_{(u,v)}$,$l_{(v)}$) , we finds the adjacent edges of e in the graph where the identifiers of the adjacent edges are the global edge identifiers in the edge dictionary. Furthermore, the edge code representation of each graph is in the form of string for further string comparisons efficiently. Figure 3 describes algorithm for generating edge code.

Algorithm GenerateEdgeCode(G_i)

Input: GDB ←{G_1,G_2,…,G_n}, EDict
Output: EC_{store}←{$EC(G_1)$, $EC(G_2)$,…, $EC(G_n)$}

∀ $D_e(G_i) \in G_i$
  Find all e_{adj} for $D_e(G_i)$
  Substitute each e_{adj} with corresponding ID in edge dictionary
  $EC(G_i)$:=all e_{adj} of $D_e(G_i)$
  EC_{store}:= EC_{store}+$EC(G_i)$
Return EC_{store}

Figure 3.GenerateEdgeCode algorithm

Table 3 shows adjacent edge information in the given graph in figure 1(c). Table 4 illustrates the detail information of adjacent edges for each edge in the graph with corresponding edge ID in Edge Dictionary.

Table 3. Adjacent edge information of (G_{Para})

Edge ID in dictionary	Edge	Edge ID in the graph
1	H,s,O	1
2	O,s,C	2

3	C,d,C	3,5,7
4	C,s,C	4,6,8,12
5	C,s,N	9,11
6	H,s,N	10
7	C,d,O	13

Table 4. Adjacent edge information of (G_{Para}) using IDs in edge dictionary

Edge ID in the graph	Adjacent edge information
1	{2}
2	{1,3,4}
3	{2,4,4},{4,4,5},{4,4}
4	{3,3},{3,3,5},{3,3,2},{7,5}
5	{3,4,5,6},{7,4,5,6}
6	{5,5}
7	{4,5}

Therefore, the edge code of the graph G_{Para} is defined in term of string and describes as follows.

$EC(G_{Para})$=1{2},2{1,3,4},3{{2,4,4},{4,4,5},{4,4}},4{{3,3},{3,3,5},{3,3,2},{7,5}},5{{3,4,5,6},{7,4,5,6}},6{5,5},7{4,5}

5.3. Graph Matching Engine

The proposed system supports two types of queries over chemical compound graph databases, namely, exact graph matching and graph similarity searching. By inputting a query graph based on exact graph matching, we can identify those molecules in the database that is symmetric to the query graph. On the other hand, given a query graph Q and a minimum similarity value sim_{min}, graph similarity searching or approximate graph matching finds all molecules in the database whose similarity values with Q is greater than or equal to sim_{min} based on the edge code representation of Q and the database graphs.

Our proposed system relies on the properties of edge codes over chemical graphs during query processing. The edge code is represented as string so that operations on strings can be performed in linear time.

6. Exact Graph Matching over Chemical Compound Database

First**,** the idea of our proposed system is to verify the substances contained in the drug that have the sufficient amount of chemical elements in definite proportion by weight as described in section 5. The proposed algorithm for exact graph matching is described in figure 4. Figure 5 shows the example illustration of exact graph query in chemical compound database.

Algorithm ExactGraphSearch(G_q)

result:= ∅
$EC(G_q)$:=GenerateEdgeCode(G_q)
∀ $EC(G_i) \in EC_{store}$
  If $|G_q| = |G_i|$ then
    If $EC(G_q) = EC(G_i)$ then
      result:=G_i
    End if
  End if
Return result

Figure 4.ExactGraphSearch algorithm

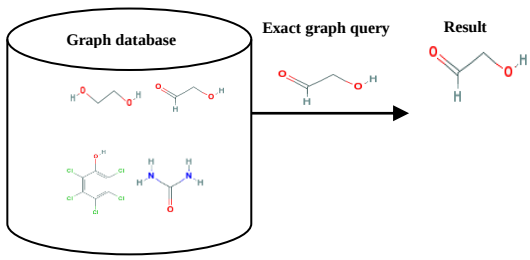


Figure 5.An example illustration of exact graph query in chemical graph database

7. Graph Similarity Searching over Chemical Compound Database

Second, other fact of our proposed system is to search chemical similarity. Chemical similarity is also one of the most important concepts in chemical informatics. It plays an important role to predicting the properties of chemical compounds, designing chemicals with a predefined set of properties and especially, in conducting drug design studies by screening large databases containing structures of available chemicals. Similarity compounds have similar properties.

Fuzzy query string search method is used to find the similarity compounds for the given graph query Q. The fuzzy similarity measurement is based on the Levenshtein edit distance (LED) algorithm. The Levenshtein distance is a string metric for measuring the amount of difference between two sequences. The distance *d* between two strings is the minimum number of edits needed to transform one string to other, with the allowable edit operations such as insertion, deletion, or substitution of a single character. The Levenshtein matrix is used to compare two strings for similarity. String similarity increases as LED decreases. Measuring string similarity with LED is precise. To find the similarity between two strings, we define the similarity formula as follows:

$$Sim_{edit}=1/(1+d(x,y)) \quad (1)$$

Where d(x,y) is the minimum number of operations needed to transform one string to another. The Sim_{edit} is in the range between 0 and 1.The algorithm for graph similarity searching is described in figure 6. An illustration example of chemical similarity searching is demonstrated in figure 7.

Algorithm GraphSimilaritySearch(Gq, sim_{min})

resultset:= ∅

$EC(G_q)$:=GenerateEdgeCode(G_q)

∀ $EC(G_i) \in EC_{store}$

  If $|G_q| = |G_i|$ then

    distance=d($EC(G_q)$, $EC(G_i)$)

    sim_{edit}=1/(1+distance)

    If $sim_{edit} >= sim_{min}$ then

      resultset:=resultset+G_i

    End if

  End if

Return resultset

Figure 6.ExactGraphSearch algorithm

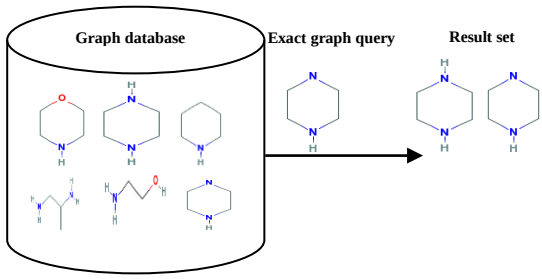


Figure 7.An example illustration of graph similarity query in chemical graph database

For the example query in figure 7 with sim_{min} =0.75, the system retrieves the database graphs if their sim_{min} is greater than or equal to 0.75.

8. Experimental Results

The performance of the generating edge codes is tested on different types of chemical graphs such as sparse, dense and complete graphs. We experienced our proposed work using chemical compound dataset from http://pubchem.ncbi.nlm.nih.gov/. Then graph index construction time was measured for different types of graphs in second. All experiments were made using a 3GHz Intel Core 2 Duo CPU with 1 GB memory and Microsoft Windows XP.

Figure 8 shows a comparison of graph index construction time for three types of graphs: sparse, dense and complete graphs. The results are obtained using chemical graph data sets by varying the graph size from 10 to 25. From the empirical analysis, it is found that the time of execution varies for sparse, dense, and complete graphs. This is because our proposed work is based on edge based representation. It takes considerable time for complete graphs when compared to sparse and dense graphs.

Figure 9 shows the querying time of exact graph matching over different query sizes. The edge codes of the graphs are strings and the comparisons between strings are proficient. Moreover, the average querying times is significantly less than a second. Figure 10 describes the exact graph matching times over different database sizes varying from 100 to 500 graphs. The graphs contained in the database are the average number of sparse, dense and complete graphs with average graph size 15. We

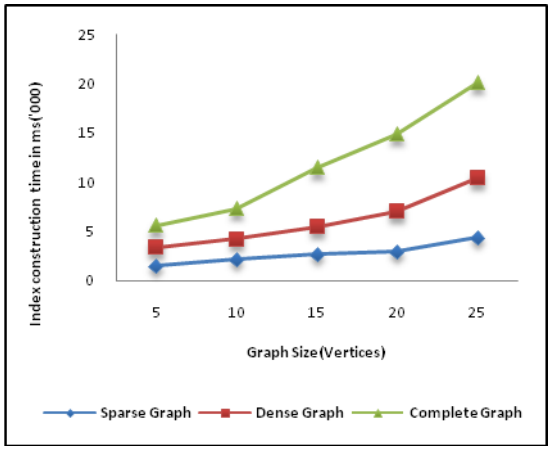


Figure 8. Graph index construction time for different types of graphs

tested the graph matching times using the query graph size 15. Although the index generating time over different graphs is significantly high, the query processing time over different graphs is less than a second.

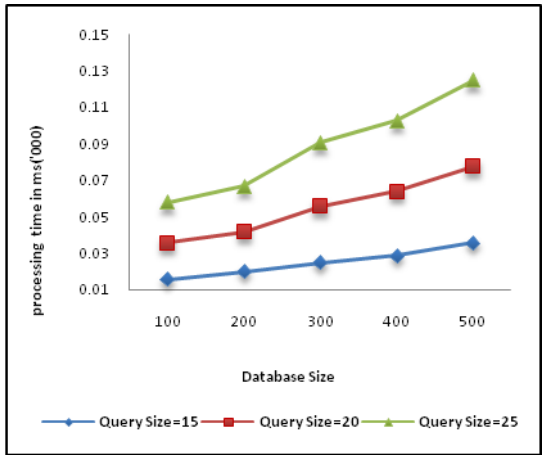


Figure 9. Exact graph querying time over different query sizes

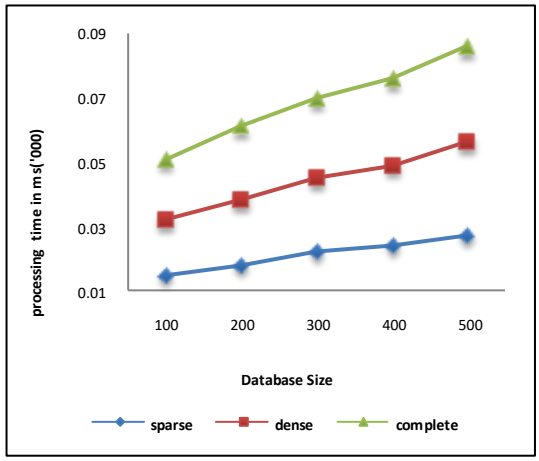


Figure 10. Exact graph querying time for different types of graphs (query size=15)

Figure 11 shows the querying time of similarity graph searching over different query sizes including 15, 20 and 25 with sim_{min}=0.75. The execution time for both types of queries is based on the edge codes in term of strings. However, the execution time for similarity graph searching is slightly higher than exact graph matching. Figure 12 shows the execution time over various database sizes. Chemical graphs were obtained randomly from the set of all

molecular structures that are represented in the dataset with average size 15.

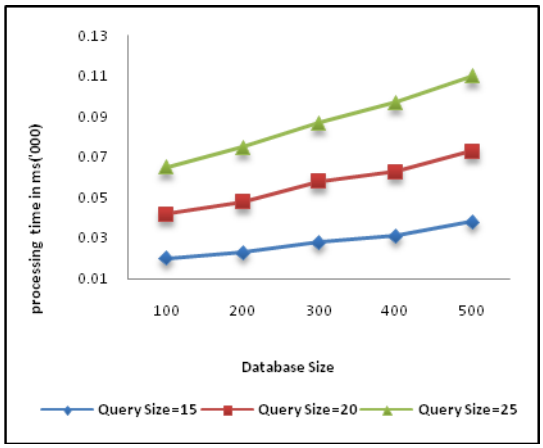


Figure 11. Similarity graph querying time over different query sizes (sim_{min}=0.75)

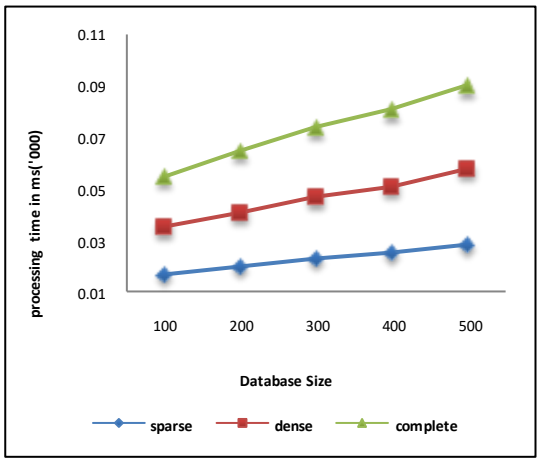


Figure 12. Similarity graph querying time for different types of graphs (query size=15, sim_{min}=0.75)

From our experimental result, the execution time is more effective and efficient for sparse and non-sparse graphs rather than complete graphs with irrespective of vertices in the graphs because our proposed work is based on edge-based representation.

9. Conclusion

Our goal is to find similarity and exact structures of chemical compound using graphs more efficiently. The proposed system uses the edge dictionary to generate the edge code. The edge code incorporates the structural information of chemical graph. In this paper, we develop the system for efficient exact graph matching and graph similarity searching using the edge code. The edge code improves the precision of exact graph matching. From the experimental result, the querying time is less than a second using the edge code. Moreover, similarity graph searching using fuzzy string matching also improves the computational speed. Our proposed system is more effective for molecular compounds where the potential drug formulas are automatically tested for a desired activity in faster query processing time.

References

[1]A. Gide, "The Structure of Matter and the Chemical Element", 2007.

[2]C. Borgelt, M. R. Berthold,"Mining Molecular Fragments: Finding Relevant Substructures of Molecules", 2003.

[3] D. Pal and P.R. Rao, "A Tool for Fast Indexing and Querying of Graphs", ACM, India, April 2011.

[4]D. Pavlov and I. Shturts, "Chemical Substructure Search Screening with Fingerprints Built with Subgraph Enumeration", 2009.

[5]D. Lin, "An Information-Theoretic Definition of Similarity", University of Manitoba, 2007.

[6]G. D. Fatta and M. R. Berthold, " High Performance Subgraph Mining in Molecular Compounds", Springer's LNCS Proc. Of the 2005 Int. Conf. on High Performance Computing and Communications, Sorrento, Italy, September 21-24, 2005.

[7]Levenshtein," The Levenshtein-Algorithm".

[8] R. Vijayalakshmi, R. Nadarajan, P. Nirmala and M. Thilage, "A Novel Approach for Detection and Elimination of Automorphic Graphs in Graph Database", Int. J. Open Problems Compt. Math., Vol. 3, No. 1, March 2010.

[9]S. Sakr and G. AI-Naymat, " An Efficient Features-Based Processing Technique for Supergraph Queries", ACM, 2010.

[10]W. Wang, C. Wang, Y. Zhu, B. Shi, J. Pei, X. Yan and J. Han, "GraphMiner: A Structural Pattern-Mining System for Large Disk-based Graph Databases and Its Applications", ACM, 2005.

[11] X. Wang, J. Huan, A. Smalter and G.H. Lushington, "Application of Kernel Functions for Accurate Similarity Search in Large Chemical Databases", International conference on Bioinformatics and Biomedicine, IEEE, Washington DC, USA. November 2009.

[12] X. Yan, P.S. Yu and J. Han, "Substructure Similarity Search in Graph Databases", 2005.