MATCH
Communications in Mathematical and in Computer Chemistry

MATCH Commun. Math. Comput. Chem. **77** (2017) 273-296

ISSN 0340 - 6253

Centrality Measures based Algorithm for Computing a Maximal Common Connected Edge Subgraph of Two Chemical Graphs

Lekshmi Ramasubramony Sulochana*, **Nirmala Parisutham, Nadarajan Rethnasamy**

Department of Applied Mathematics and Computational Sciences,
PSG College of Technology,
Coimbatore 641 004, Tamilnadu, India
rsl@mca.psgtech.ac.in

(Received April 1, 2016)

Abstract

Graph similarity plays an essential role in various fields of scientific and industry oriented applications such as bioinformatics and computational chemistry. The process of determining structural similarities between chemical structures of molecules helps to identify common behavior of these molecules. A promising approach to capture the structural similarity between two chemical compounds is to detect a maximal Common Connected Edge induced Subgraph (CCES) in their molecular graphs. This paper detects a large sized maximal CCES of two given chemical graphs using a new technique incorporating centrality measures. The idea is to use a DFS search tree whose root node is chosen as the one having the highest average of closeness and reach centrality measures from the tensor product graph of the input graphs. This measure of average narrows down the search space of the problem. The experimental results, on synthetic and real chemical database, further ensure the efficiency of the proposed algorithm when compared with the existing works.

1 Introduction

Molecules consist of atoms and the bonds linking them. There are many ways to represent a molecule. The chemical formula is one of the ways of expressing a molecule; it describes the composition of elements. For example, a carbon atom has four valences, which can be

used to form various stable bonds with other atoms, such as carbon, oxygen, and nitrogen. In computational chemistry, the chemical formula is not much helpful in revealing the bonding and structural information about the molecule, since two different compounds with the same chemical formula can exist. For example, C_2H_6O is for dimethyl ether and ethanol; these molecules are called isomers. Hence, a promising way is needed to express each chemical compound uniquely. Such a need is satisfied by graphs, which are powerful ways of representing chemical molecules, called chemical graphs. A compound can be easily visualized as an undirected labeled graph in which atoms are nodes with labels representing the name of the atoms, and bonds are the edges. For example, Figure 1 depicts two chemical graphs of 2-Methylaziridine (C_3H_7N), and Pyrimidine ($C_4H_4N_2$). One of the basic widely accepted principle in chemistry is that compounds with similar structures frequently share similar physicochemical properties and biological activities [1–3]. This principle has been successfully applied in various areas, such as 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 (or potentially available) chemicals and enzyme-promiscuity prediction, in which data from experimentally measured compounds is used to make prediction about the properties of the compound currently being investigated [4,5]. Because of the importance of the principle in drug discovery research, many similarity measures have been proposed to accurately quantify the similarity between compounds.

Computer algorithms for chemical search methodology are called virtual screening, the idea being that these methods are testing large numbers of compounds by computer instead of by experiment [6]. The prediction of biologically active compounds is of great importance for virtual screening approaches in drug discovery and chemical genomics [4,5].

One of the most applied concepts in finding similarity between compounds is maximal Common Connected Edge induced Subgraph (CCES). Finding maximal CCES in measuring similarity of chemical structures has numerous advantages [7,8]. First, the largest common substructure of structurally related chemicals is likely to be an important component of their similar biological activities; second, the common pattern can be envisaged by considering the common subgraph between two chemical structures [4].

Many existing approaches consider the maximum CCES by reducing it to the problem of finding the maximum clique of constructed association graph. Both, maximum common

subgraph and maximum clique detections, are NP-complete problems [9–11]. A study of such algorithms, together with an analysis of their complexity, and potential applications is discussed in [12].

The aim of this work is to find a large sized maximal CCES of two chemical graphs with the aid of two centrality measures - closeness and reach. Centrality measures determine the relative importance of a vertex in a graph [13–15]. The above said two measures are based on shortest path distance of each node to every other node of the graph. At first, a tensor product graph is constructed from the two input graphs. Then, closeness and reach centralities are used to select a node with high closeness and reach value from the tensor product graph, and that node is considered as an initial node to start DFS search on the tensor product graph to detect a maximal CCES. The proposed method is guaranteed to give at least one maximal CCES which may or may not be a maximum CCES. The detailed construction will be explained later.

The paper is organized as follows. Essential definitions are provided in section 2. An overview of the existing algorithms to find maximal CCES is discussed in section 3. The proposed algorithmic solution for finding a large sized maximal CCES is discussed in section 4. The time complexity and performance of the proposed algorithm are discussed in section 5. Section 6 provides an extensive experimental evaluation of the proposed algorithm on synthetic and real chemical graph database. Section 7 draws some conclusions and throws light on future direction of graph similarity.

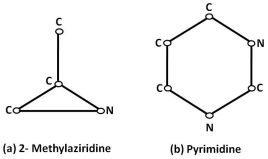

Figure 1: Chemical graphs

2 Definitions and notations

In this section, a few basic concepts and definitions used throughout the paper will be introduced. [16] is mainly followed for the concepts in graph theory.

Undirected graph: An undirected graph G consists of two sets V_G whose elements are called vertices (or nodes), and E_G, the elements of which are called edges. Each edge

has a set of one or two vertices associated to it, which are called its end vertices. An edge $e = (v_1, v_2)$ is said to join its end vertices. If the ends of an edge e are the same, then e is a self-loop. If edges e_1 and e_2 have the same end vertices then they are said to be parallel. A simple graph is a graph that has neither self-loops nor parallel edges. The degree of a vertex v in a simple graph is the number of edges incident on it and is denoted as $degree(v)$.

Labeled Graph: A labeled graph is a graph with labels, typically v_1, v_2,$\cdots$, v_n, assigned to the vertices. Two labeled graphs with the same set of labels are considered the same only if there is an isomorphism from one to the other that preserves the labels and the adjacency.

Subgraph: A subgraph of a graph G is a graph H such that $V_H \subseteq V_G$ and $E_H \subseteq E_G$.

Edge Induced Subgraph: An edge induced subgraph is a subset of the edges of a graph G along with all their end vertices.

Common Connected Edge Induced Subgraph: If two connected edge induced subgraphs G_{s1} and G_{s2} of G_1 and G_2 respectively are such that G_{s1} is isomorphic to G_{s2} then $G(= G_{s1}$ or $G_{s2})$ is called a common connected edge induced subgraph of G_1 and G_2. Common Connected Edge induced Subgraph is abbreviated as CCES.

Maximal Common Connected Edge Induced Subgraph: A common connected edge induced subgraph G_3 of G_1 and G_2 is called a maximal common connected edge induced subgraph, if there is no other common connected edge induced subgraph H, such that $G_3 \subset H$. An example for maximal CCES is shown in Figure 2.

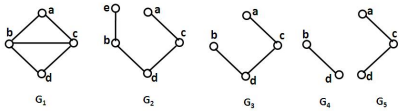

Figure 2: For G_1 and G_2, G_3 is a maximal CCES

Edge Density: For a graph G with $|E|$ edges and $|V|$ vertices, the edge density $EdgeDen(G)$ is the ratio of the number of edges in the given graph to the maximum possible number of edges.

In graphs, a natural distance metric $d(v_i, v_j)$ can be defined between all pairs of nodes v_i and v_j as the length of their shortest paths. The eccentricity of a vertex v_i denoted as $e(v_i)$ is $\overset{max}{p} d(v_i, v_p)$. The farness of a node is defined as the sum of its distances to all

other nodes, and its closeness is defined as the reciprocal of the farness.

The following centrality measures are used in the proposed work to find a maximum common connected edge subgrah.

Closeness Centrality: The closeness centrality $CC_G(v_i)$, of a vertex v_i in a graph G with n nodes is measured using the formula,

$$CC_G(v_i) = \frac{n-1}{\sum_{p=1,p\neq i}^{n} d(v_i,v_p)}$$

The more central a node is the lower its total distance to all other nodes.

Reach Centrality: The reach centrality $RC_G(v_i)$, of a vertex v_i in a graph G is measured using the formula,

$$RC_G(v_i) = 1 + \sum_{x=1}^{e(v_i)} \frac{r_{ix}}{x}$$

where r_{ix} is the number of vertices at x hop distance from v_i. The upper limit in the summation is due to the fact that the maximum hop a node can have is its eccentricity.

Tensor Product Graph: In graph theory, the tensor product $G \times H$ of graphs G and H is a graph such that the vertex set of $G \times H$ is the Cartesian product $V(G) \times V(H)$; and any two vertices $u : u'$ and $v : v'$ are adjacent in $G \times H$ if and only if u is adjacent with v and u' is adjacent with v'. The tensor product of two graphs is commutative.

Depth First Search: It is a systematic method of visiting the vertices of a graph G. The search starts with any vertex u of G, and extends its search to vertex adjacent to u which has not yet been visited. If no such vertex exists then it returns to the vertex visited just before the current node and the search is repeated until every vertex of G has been visited.

In this paper, all graphs considered are simple, undirected and vertex labeled, unless otherwise specified.

3 Related works

The prediction of biologically active compounds is an important process in drug discovery and chemical genomics [17]. There are many research papers involved in determining the similarity of the chemical compounds to predict active compounds. Few such benchmark methods are discussed in this section.

One of the familiar benchmark algorithms is Durand-Pasari algorithm that is based on association graph and clique detection in it [18,19]. The vertices of the association graph correspond to pairs of vertices of the two input graphs having similar attributes. The

conversion makes it much harder to detect Maximum Common induced Subgraph (MCS) effectively when the size of the graph database is large. Cliques in the association graph of G and H correspond to isomorphisms of induced subgraphs of G and H. Thus, the association graph can be used to reduce the problem of induced subgraph isomorphism to the problem of finding cliques in association graphs [9, 10]. Specifically, the largest graph that is an induced subgraph of both G and H corresponds to the maximum clique in their association graph. Although the problems of finding largest common induced subgraphs and of finding maximum cliques are NP-complete, this reduction allows clique-finding algorithms to be applied to the common subgraph problem [10, 20].

Yiqun Cao et al. [4] developed a backtracking algorithm to predict MCS of two input graphs. It generates a search tree where MCS are stored only in the leaf nodes of the tree. It has to explore all intermediate nodes, in each step there is a heuristic checking for induced subgraph. The size of the search tree is directly proportional to the size of the input graphs. Berlo et al. method [5] uses a kind of association graph to find all maximum common connected induced subgraph and also maximum common connected edge subgraph. The weight of each node is computed by summing the shortest path with every other node. The matching process is started with the node which has the highest weight and other nodes are followed based on their weights to find MCS. The matching process has been done on all nodes in the association graph and the stopping point criteria is made only after checking all nodes including those nodes which are not contributing to the construction of MCS. Cao et al. method is efficient for small size MCS and when the size of the MCS gets large, Berlo et al. method is appreciated [5]. However both methods explore all possible connections of all nodes in order to obtain all maximum CCES of two input graphs.

Betweenness centrality measure has been used in [21], to predict tandem structural repeats in proteins, by identifying the nodes with high betweenness value. However, this method gives only approximate structural matches in proteins.

Unlike most existing approaches, the proposed work tells in a determinant way of selecting a node which serves as an initial node of the DFS search to predict a large sized maximal CCES. The novelty of this work is that it does not deal with all unwanted nodes which are not contributing to detect a maximal CCES.

4 Proposed work

A chemical molecule could be represented as an undirected, labeled graph in which atoms are nodes and bonds are the edges of the graph. Chemical graphs are easy to handle in the sense that vertex degree is bound to a small constant (normally at most 4), and the size of each graph is reasonably small (typically around 20-25 edges) [1]. A database with only chemical graphs is called a chemical graph database. In this paper, chemical graph database is analyzed to predict a kind of common pattern known as maximal Common Connected Edge induced Subgraph (CCES) to characterize the molecules stored in the database.

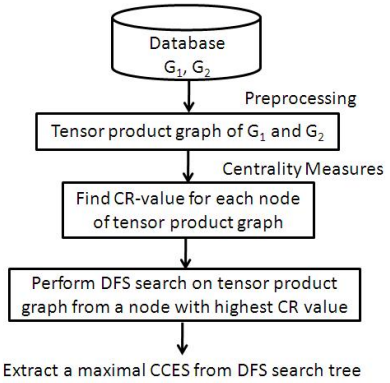

Figure 3: The phases of proposed maximal CCES algorithm

Finding all maximal CCES is an NP-complete problem [10]. This paper coins an idea which finds at least one large sized maximal CCES of a pair of chemical graphs. The high level view of the proposed idea is outlined as follows:

- Given a pair of chemical graphs G and H, first find their tensor product graph and denote it as $G \bigotimes_A H$. Using closeness and reach centralities, a non negative value named CR-value is calculated for each node of $G \bigotimes_A H$.
- Then a DFS is performed on $G \bigotimes_A H$ and a search tree is constructed starting with a node as the one with the highest CR-value.

This feature promises the time complexity in polynomial and guarantee at least one maximal CCES. The two chemical compounds 2-Methylaziridine and Pyrimidine shown

in Figure 1 are considered as example of the input graphs for the proposed algorithm to find maximal CCES.

The various phases involved in the proposed algorithm in detecting a maximal CCES are shown in Figure 3.

4.1 Preprocessing

In preprocessing phase, each atom of a chemical graph is labeled uniquely, in order to distinguish same atoms while processing them to predict a maximal CCES. For convenient identification, the atomic symbol of the atom is used while naming these atoms, for instance carbon atoms are represented as $C_1, C_2, \ldots, C_i$. The chemical graph database shown in Figure 1, is uniquely labeled and depicted in Figure 4.

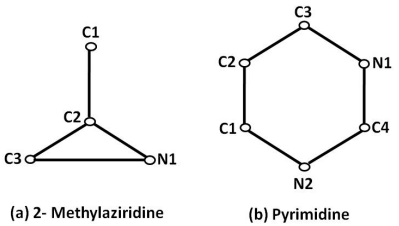

Figure 4: Chemical graphs with unique labels

Let G and H be labeled chemical graphs whose tensor product graph $G \bigotimes_A H$ is a graph such that the vertex set of $G \bigotimes_A H$ is $V(G) \times V(H)$ and any two vertices $u : u'$ and $v : v'$ are adjacent in $G \bigotimes_A H$ if and only if u and $u^{'}$ represent same atomic symbol in G and H respectively, similarly $v : v'$; and there are edges (u, v), (u', v') in G and H respectively. This is a restricted form of the tensor product. The range of the number of vertices of tensor product graph is $1 \leq i \leq (|V_{G_1}| \times |V_{G_2}|)$. In other words, same atoms of two input graphs are only combined to form a node in the tensor product graph. Each node of the tensor product graph is represented as k-v_i:v_j where k is a positive integer, $1 \leq k \leq (|V_{G_1}| \times |V_{G_2}|)$, v_i and v_j are the labels of the vertices in G_1, and G_2 respectively (- symbol is a hyphen and not to be confused as minus). For example, the tensor product graph of 2-Methylaziridine and Pyrimidine is shown in Figure 5, the vertex 7-C2:C3 denotes that C2 is a vertex of 2-Methylaziridine, C3 is a vertex of Pyrimidine and the node C2:C3 is a seventh node of the tensor product graph. The tensor product graph has fewer vertices when the atoms in G are much different from that of H. If the

atoms of G and H are different, i.e. no atom is common for G and H, and then the tensor product graph of G and H is a null (empty) graph.

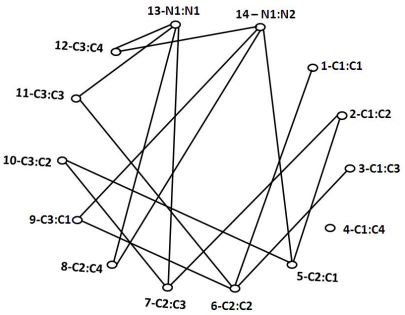

Figure 5: Tensor product graph of the graphs (a) and (b) shown in Fig. 4

The tensor product graph constructed as above is processed further to find a maximal CCES, which is discussed in the next section.

4.2 Proposed CR-maximal CCES Algorithm

Centrality measures are helpful in determining the relative importance of a vertex in the graph, and there are many centrality measures, among these, closeness, and reach centralities are used here to analyze the relative importance of each node of the tensor product graph. This knowledge is helpful to reduce the search space of the problem.

The search can be started from a node with high relative importance in the tensor product graph. Instead of starting the search from all nodes, it is sufficient to start from a node which is central to other nodes, identifying a large sized maximal CCES. This algorithm ensures at least one maximal CCES; more than one may also be generated.

Calculation of CR-value

Closeness and reach centralities of a node v_i are calculated based on the shortest path from v_i to every other nodes of the tensor product graph, and deal with how well each node is close to all other nodes in a graph. Even though closeness and reach centrality measures are useful in their own sense, a single reference value of centrality will be helpful. Hence a reasonable consideration is to take the average of the two measures.

The average closeness - reach centrality value (CR-value) of a vertex v_i is calculated as follows,

Table 1: CR-values of nodes in the tensor product graph shown in Figure 5

$Node$	$Closeness$	$Reach$	$CR-value$
C1:C1	0.24528	0.398	0.32164
C1:C2	0.27083	0.463	0.366915
C1:C3	0.24528	0.398	0.32164
C2:C1	0.30233	0.530	0.416165
C2:C2	0.30952	0.560	0.43476
C2:C3	0.30233	0.530	0.416165
C2:C4	0.30233	0.500	0.401165
C3:C1	0.31707	0.518	0.417535
C3:C2	0.27083	0.463	0.366915
C3:C3	0.31707	0.518	0.417535
C3:C4	0.30233	0.500	0.401165
N1:N1	0.34211	0.595	0.468555
N1:N2	0.34211	0.595	0.468555

$$\text{CR-value } (v_i) = (closeness(v_i) + reach(v_i))/2$$

The nodes with highest CR-value can serve as initial nodes in the process of DFS search for finding maximal CCES.

Table 1 depicts the closeness centrality, reach centrality and CR-value of the tensor product graph shown in Figure 5. The values are calculated using the analytical tool UCINET 6 for Windows 6.508.

The following are the steps of the proposed CR-maximal CCES algorithm to find a large sized common connected edge subgraph using CR-value.

Step 1: CR-value of each vertex of the tensor product graph is calculated.

Step 2: Any one of the nodes with the highest CR-value is selected as a root for the DFS search, which can be visualized as a DFS search tree.

In each level of DFS search tree, we find (i) potential neighbors as candidate sets for the nodes at that level and then (ii) group the neighbors based on a specific rule. To make the understanding easier we consider the tensor product graph in Figure 6 as our running example.

(i)Finding candidate sets and grouping:

Level 1:

If there is a unique vertex with the highest CR-value, it is chosen as the root of the DFS search tree. If there are many vertices with the highest CR-value, then an arbitrary one is selected as the root. The maximum number of children of the root is the number

of neighbors of the root in the tensor product graph. A grouping process is carried out among the neigbours of the root and each group of the nodes is a child of the root. The process is as follows,

Let the neighbors of the root be $l_i : l_j$ where $1 \leq i \leq |V_{G_1}|$, $1 \leq j \leq |V_{G_2}|$.

Consider two neighbors $l_p : l_q$, and $l_m : l_n$, $p \neq m$ and $q \neq n$, then $l_p : l_q$, and $l_m : l_n$ are accommodated in the same group. This is verified for all the neighbors of the root, and suppose k_1 groups are identified then each group becomes a child of the root. For instance, there are two nodes N1:N1, and N1:N2 with same highest CR-value, and can be seen in Table 1. Arbitrarily, the node 14-N1:N2 is selected from the tensor product graph shown in Figure 6 as the root node of DFS search tree, and it has four neighbors 5-C2:C1, 8-C2:C4, 9-C3:C1 and 12-C3:C4. These neighbors are grouped as {5-C2:C1, and 12-C3:C4}, and {8-C2:C4, and 9-C3:C1} based on the above grouping rule. Hence, these two are the children of the root node 14-N1:N2. The pictorial representation of a DFS search tree with 14-N1:N2 as root node is shown in Figure 6. We cannot allow one node in one graph to correspond to multiple nodes in the other graph. For example, 5-C2:C1 is not grouped with 8-C2:C4, since C2 of 2-Methylaziridine corresponds to C1 and C4 of Pyrimidine. That is, while doing DFS search, at any point of time, the one to one correspondence should be maintained to reach a valid maximal CCES.

Leve1 i (i>1):

Each child of the nodes at level i-1 consists of either a single node or a collection of nodes of the tensor product graph. Suppose there are k_{i-1} nodes (children) at level i, then neighbors of each child is identified, and grouped according to the rule above to get their children. In principle, while expanding each branch, a node occurred already is not allowed again to be part of that branch (it may or may not occur in other branches).

(ii) Stopping Criteria:

In the process, a node becomes a leaf node, when all the neighbors of that node have already occurred in that branch.

For example, in Figure 6, the neighbors of right node of Level 2 are 13-N1:N1, 14-N1:N2, and 6-C2:C2. The nodes of the right branch are N1:N2, C2:C4, and C3:C1. N1 and C2 of 2-Methylaziridine are already matched with N2 and C4 of Pyrimidine respectively, so that there are no new correspondence nodes to proceed the branch further.

Notice that the DFS tree need not always contain all nodes of the tensor product

graph, so that there may be unvisited nodes of the tensor product graph. Maximum number of levels of a DFS search tree of tensor product graph is the diameter of the tensor product graph.

Step 3: Once DFS search tree construction is over, the following are performed.

Suppose the height of the tree is h and there are m leaves; then there are m branches of the tree. Each branch of the tree consists of the set of nodes from the root to the leaf. There are m maximal CCES and it is possible for a node to occur in multiple branches, since two or more child nodes may have the same parent node. We recognize the corresponding adjacent edges of these nodes from the two given input graphs and such possible subgraphs of the input graphs are collected. From the DFS search tree shown in Figure 6, two subgraphs can be identified, and they are shown in bold in Figure 7. Among these two, subgraph (i) is declared as a large sized maximal CCES, since it has the highest number of edges when compared to the other.

From these subgraphs, identify the subgraph(s) with the highest number of edges; which is/are our required maximal CCES.

The maximum number of maximal CCES obtained from a DFS search tree is the number of leaf nodes of that tree. The objective of the proposed work is to find a maximal CCES and for this, it is sufficient to establish any one of the branches of the search tree. BFS search will be applicable if all branches of the tree should be established, hence not preferred for our method.

CR-value is not calculated for the nodes which are isolated in the tensor product graph. In the running example 4-C1:C4 is an isolated node, hence CR-value for that node is not determined.

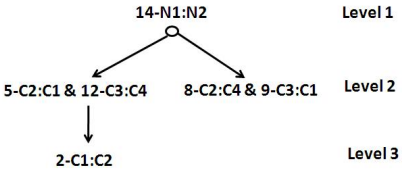

Figure 6: DFS Search tree of the tensor product graph shown in Figure 5

To reason about the optimality of the algorithm, we first argue that the center of a graph and a node with the highest CR-value in it need not to be same. According to the definition of closeness centrality, a node with the lowest sum of shortest paths has the

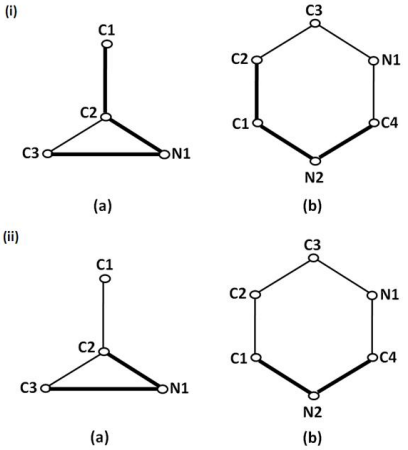

Figure 7: In (i) and (ii), thick lines depict the Common Connected Edge induced Subgraphs of (a) 2-Methylaziridine (b) Pyrimidine. (i) and (ii) are the subgraphs identified from DFS search, from these, (i) is identified as a large sized maximal CCES.

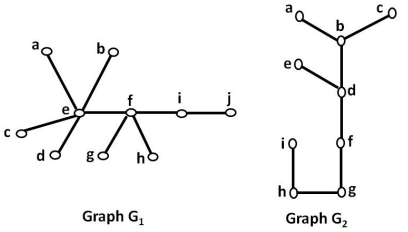

Figure 8: Graphs G_1 and G_2

highest closeness centrality, whereas a node with the lowest eccentricity is a center node of the graph.

For the graph G_1 in Figure 8, f is the center, f and e have the highest closeness centrality value, and e has the highest reach centrality value, and hence e has the highest CR-value. However, e is not a center of G_1.

For G_2, d has the highest closeness as well as reach centrality values, hence the highest CR-value, whereas the center is f. This gives a clear indication that, a node with the highest CR-value in a graph need not be a center of the graph. From G_2, it is possible to note that a node with the highest degree need not be a node with the highest closeness centrality. There are graphs in which a node with highest reach centrality need not be

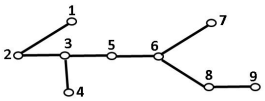

Figure 9: Graph G

the one with highest closeness centrality. For example, graph G in Figure 9, node 5 has the highest closeness centrality whereas nodes 3 and 6 have highest reach centrality.

CR-value of all the nodes of an tensor product graph of G and H is same when G and H are complete graphs. G and H may have same or different number of vertices, with all the nodes indicating same atom. In the complete graph, K_n where n is the number of vertices, each vertex has n-1 neighbors. Hence the normalized closeness centrality = reach centrality = 1. Therefore each vertex has the same CR-value, and every vertex has the same relative importance with all other vertices of the graph.

Many chemical graphs are cycles. For a tensor product graph of two cycle graphs with different sizes, every vertex has (i) the same closeness centrality, (ii) the same reach centrality. If G and H are cycle graphs then $G \bigotimes_A H$ becomes 4-regular. For this regular graph, all nodes have the same sum of shortest path distance with other nodes, and each node has same eccentricity value. Hence all nodes have same closeness centrality and reach centrality values.

4.2.1 Algorithm

The overall structure of the algorithm is as follows,

Preprocessing Algorithm

Input: Graphs G and H

Output: Tensor product graph $G \bigotimes_A H$

1: Assign unique label to each node of G and H

2: Construct tensor product graph of input graphs G and H

CR-maximal CCES Algorithm

Input: Tensor product graph $G \bigotimes_A H$

Output: A maximal CCES

1: CR-value is calculated for each node of the tensor product graph

2: A node v_{i_G}:v_{j_H} with the highest CR-value is selected as the root node of the DFS search tree from the tensor product graph

3: For each step of the DFS search, collect all neighbors of the current node v_{i_G}:v_{j_H} in branch k of the DFS search tree

4: if neighbors of v_{i_G}:$v_{j_H} \neq \phi$ and all the neighbors of that node have not already occurred in that branch then

5: Combine the collected neighbors of the current branch, in order to, not allowing a node in one graph to correspond to multiple nodes in the other graph.

6: else //current node v_{i_G}:v_{j_H} is a leaf node of
DFS search tree

7: expand next branch; k = k+1

8: end For

Subgraphs of each branch of DFS search tree are identified, then the subgraph with maximum number of edges is declared as a large sized maximal CCES.

4.3 Proof of correctness

This section explains the proof of correctness of the proposed algorithm and a few related results.

Lemma 1. A DFS search tree of the tensor product graph of G and H with root node as one of the nodes having the highest CR-value of the tensor product graph is sufficient to get at least one maximal CCES of G and H.

Proof. A node with the highest CR-value indicates that it has more links with other nodes of the graph, compared to the other nodes.

The resultant subgraph is connected: An edge $(u : u', v : v')$ of the tensor product graph of G and H implies that (u, v) is an edge of G and (u', v') is an edge of H. Hence DFS search on $G \bigotimes_A H$ identifies only connected subgraphs of G and H.

The resultant subgraph is a maximal CCES: The CR-value of a node v_i in $G \bigotimes_A H$ is the average of its closeness and reach centralities. These values are computed based on the shortest path from v_i to other nodes of the graph. If a node v_i has the highest CR-value then v_i is an influential node, by definition of CR-value. Hence, starting a DFS search from such an influential node v_i gives the guarantee to get maximum of matched nodes and which yields a maximal CCES.

□

The algorithm suggested that the initial node of the DFS search tree should be with the highest CR-value, since the node with the highest CR-value always takes the lowest sum of shortest path distance. The construction of DFS search tree will be done in an efficient way. The closeness and reach centrality measures are based on shortest distance, closeness particularly seeks the lowest sum of shortest path distance, whereas reach centrality seeks the value with respect to eccentricity. In many cases, reach centrality alone is sufficient to select a node to serve as an initial node of DFS search tree, to strengthen the selection process both measures are considered to identify the initial node.

Lemma 2: If graphs G and H have unique centers called x and y respectively, and the sum of shortest path distance of x and y are the lowest value compared to other nodes of G and H respectively, then a DFS search tree started from the node x:y of the tensor product graph of G and H is sufficient to find at least one maximal CCES.

Proof. The well known truth, if $e_{G \bigotimes_A H}(x : y) = radius(G \bigotimes_A H)$ then the node x:y is the center of the graph $G \bigotimes_A H$ [22]. If the eccentricity of the node x:y in $G \bigotimes_A H$ is equivalent to the radius of the graph $G \bigotimes_A H$, then such a node x:y becomes the center of the graph $G \bigotimes_A H$. When x, y are unique centers of G and H respectively, the node correspondence x:y is the only node which has low sum of shortest path distance with other nodes of $G \bigotimes_A H$, and also low eccentricity value. Hence such a node x:y has the highest closeness centrality, reach centrality and also center of the tensor product graph. □

For situations in Lemma 2, there is no need of CR-value calculation. In our observation, the node with the highest reach centrality has the lowest sum of shortest path distance. Hence, the node with the highest reach centrality is one among the nodes with the highest closeness value.

5 Computational time complexity

Three phases of computations are involved in finding the total time complexity of CR-maximal CCES algorithm. The number of comparisons needed in each phase and each of its time complexity are explained below.

Tensor product graph Chemical graphs are undirected by nature with no self loops. Let $|V_{G_1}|$ and $|V_{G_2}|$ be the number of vertices of graphs G_1 and G_2 respectively. The number of nodes in the tensor product graph G_A is $|V_{G_1}| \times |V_{G_2}|$, and the required number of comparisons to construct the tensor product graph is at most $|V_{G_A}| \times (|V_{G_A}| - 1)/2$. Thus the time complexity is $O(n^2)$ where n is the number of nodes in the tensor product graph.

Compute CR-Value In this paper, closeness and reach centrality measures are calculated using the analytical tool UCINET 6 for Windows 6.508, which follows Freeman computations [15]. The time complexities of closeness and reach centrality measures according to Freeman are $O(n^3)$ and $O(n^2)$ respectively, where n is the number of nodes in the tensor product graph.

DFS Search Tree construction

The time complexity of the DFS Search tree depends on the nature of input graphs.

When the tensor product graph is constructed with two general graphs G_1 and G_2, the height of the DFS search tree is $min(|V_{G_1}|, |V_{G_2}|)$. Strictly, the height is the eccentricity of the root node correspondence plus 1. *i.e.* eccentricity of the root node correspondence plus $1 \leq min(|V_{G_1}|, |V_{G_2}|)$.

The concept of node correspondence grouping in each step of DFS search avoids the generation of same subgraph. At each level, the maximum number of possible nodes is $(|V_{G_1}| - 1) \times (|V_{G_2}| - 1)$. So, the total number of nodes in a DFS search tree is $min(|V_{G_1}|, |V_{G_2}|) \times |V_{G_1}| - 1 \times |V_{G_2}| - 1$. Hence the total time complexity is $O(min(|V_{G_1}|, |V_{G_2}|) \times |V_{G_1}| \times |V_{G_2}|)$.

We give below the time complexities for two particular instances of the input graphs.

Case 1: When the tensor product graph is constructed with two complete graphs G_1 and G_2.

The following characteristics are observed in the tensor product graph.

(i) The tree has only two levels.

(ii) In the second level, the number of branches is $|V_{G_1}| - 1 \times |V_{G_2}| - 1$.

(iii) Each node in second level, has $(|V_{G_1}| - 1 \times |V_{G_2}| - 1)/min(|V_{G_1}|, |V_{G_2}|)$ node branches.

By having the above knowledge, the total number of nodes in the DFS search tree is identified as $((|V_{G_1}| - 1) \times (|V_{G_2}| - 1)) + 1$. Hence the time complexity is $O(|V_{G_1}| \times |V_{G_1}|)$.

Case 2: When the tensor product graph is constructed with two cycle graphs G_1 and G_2.

The following features are observed in the tensor product graph. The height of the DFS search tree is $(min(|V_{G_1}|, |V_{G_2}|))$. Since a node in a cycle graph has exactly 2 neighbors, any node $v_i : v_j$ in the tensor product graph of two cycle graphs has a maximum of 4 possible matches. Therefore the maximum number of possible branches of DFS search tree is 4. From these, it is possible to predict the total number of nodes of the tree as $(min(|V_{G_1}|, |V_{G_2}|) \times 4$. Hence the time complexity is $O(min(|V_{G_1}|, |V_{G_2}|))$.

6 Experimental analysis

A comprehensive experimental performance study is conducted on synthetic datasets. The implementation of the proposed work has been done in MATLAB version 7.12.0.635 R2011(a), 64-bit (win 64) using Intel(R) Core(TM), 1.90GHz speed with 4GB RAM size of Windows 8 operating system.

Before getting into the implementation details of the proposed CR-maximal CCES algorithm, we first discuss the performance of existing algorithms of Cao [4], Levi [10], and Berlo [5]. Levi's algorithm deals with an association graph which is a dense graph constructed from input graphs and MCS is identified by finding maximum clique in the tensor product graph. Berlo's algorithm is especially faster if the size of the MCS is large. Cao's method uses an incremental approach to search MCS. By estimating the computation time for the method of Cao, Berlo derived that Cao's method is approximately thirty times slower than Berlo's approach. But Cao's algorithm gives better performance when the order of the graphs is less than 15 nodes when compared to that of Levi and Berlo. Cao's is a less efficient algorithm for dense graphs.

We now analyse the proposed CR-maximal CCES algorithm and compare with the above said exising algorithms. The performance of CR-maximal CCES algorithm is evaluated on large size synthetic graph datasets. Synthetic data sets have been generated using MATLAB to validate the computational complexity of the proposed CR-maximal

Table 2: Comparison of time (in seconds) taken by CR-maximal CCES algorithm with other existing algorithms on synthetic datasets

$\|V\|$	*Levi*	*Cao*	*Berlo*	$CR-maximalCCES$
10	3.87	0.38	0.876	0.0282
15	17.76	5.67	6.285	1.4361
20	32.87	7.56	8.278	3.4512
25	7730.87	9.45	10.975	3.5463
30	9276.23	188.11	13.17	6.2341
35	10821.65	302.65	19.95	7.3908
40	13431.15	345.15	22.78	10.4872
45	13914.34	225391.64	173.53	87.4310
50	367622.84	250435.16	193.17	117.2912
55	404385.12	275478.66	215.81	132.4872
60	442147.48	324786.21	237.804	145.8720

CCES algorithm.

For the experiments described in this section, we generated different size of data sets. First, using a series of randomly generated graphs, comprising of 10 to 60 nodes, the running time of Levi, Cao and Berlo is compared with the proposed algorithm, and the comparison is shown in Table 2 (We fix the edge density as 0.3 and MCS Size=60%). A graphical display of the data in Table 2 is shown in Figure 10.

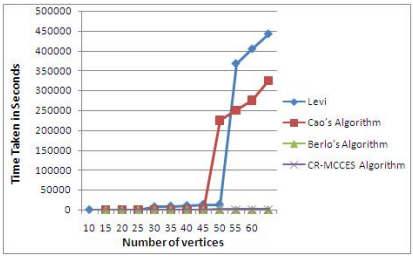

Figure 10: Comparison of running time shown in Table 2

The proposed work is closely connected with Berlo's work. The significant difference is Berlo performed DFS search on each node of the tensor product graph, whereas the CR-maximal CCES algorithm selects only one node to perform DFS search to find a maximal CCES. This feature reduces the time complexity of CR-maximal CCES algorithm drastically to yield better performance. As can be seen so far, the model of Berlo is faster when compared to all existing related works. Hence, for our second experiment, CR-maximal CCES algorithm is elaborately compared with Berlo's method by generating

Table 3: Comparison of time (in seconds) taken by CR-maximal CCES algorithm with Berlos algorithm on synthetic datasets

$\|V\|$	$BERLO$	$CR-maximalCCES$
200	792.68	243.11
300	1189.02	456.12
400	1632.92	789.15
500	1981.7	873.65
600	2378.04	1023.54
700	2774.38	1348.20
800	3170.72	1583.16
900	3867.06	1894.24
1000	4296.73	2341.78
2000	8593.46	3231.83
3000	12890.19	4129.56
5000	21483.65	7832.79

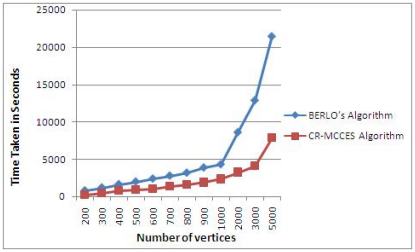

Figure 11: Comparison of Berlo algorithm with CR-maximal CCES

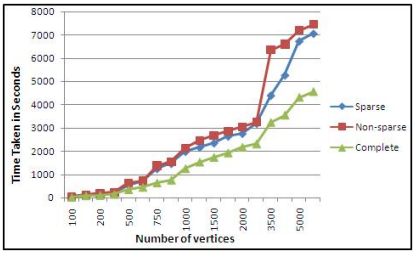

Figure 12: Comparison of the running time taken by CR-maximal CCES on sparse, non-sparse and complete graphs

Table 4: Running time of CR-maximal CCES algorithm on synthetic sparse graph datasets

$\|V\|$	$\|E\|$	β	$Seconds$
100	99	0.25	47.47
100	99	0.5	112.66
200	199	0.25	142.67
200	199	0.5	194.66
500	499	0.25	538.98
500	499	0.5	698.45
750	749	0.25	1238.45
750	749	0.5	1457.12
1000	999	0.25	1976.36
1000	999	0.5	2159.25
1500	1499	0.25	2343.67
1500	1499	0.5	2634.56
2000	1999	0.25	2756.39
2000	1999	0.5	3190.55
3500	3499	0.25	4397.23
3500	3499	0.25	5269.34
5000	4999	0.25	6734.55
5000	4999	00.5	7063.20

Table 5: Running time of CR-maximal CCES algorithm on synthetic non-sparse graph datasets

$\|V\|$	$\|E\|$	β	$Seconds$
100	2475	0.25	53.90
100	2475	0.5	123.87
200	9950	0.25	194.29
200	9950	0.5	223.45
500	62375	0.25	632.44
500	62375	0.5	724.12
750	140438	0.25	1375.46
750	140438	0.5	1534.89
1000	249750	0.25	2123.43
1000	249750	0.5	2455.23
1500	562625	0.25	2678.42
1500	562625	00.5	2859.45
2000	999500	0.25	3033.54
2000	999500	0.5	3245.11
3500	3061625	0.25	6342.68
3500	3061625	0.5	6578.12
5000	6248750	0.25	7183.20
5000	6248750	0.5	7433.55

Table 6: Running time of CR-maximal algorithm on synthetic complete graph datasets

$\|V\|$	$\|E\|$	β	$Seconds$
100	4950	0.25	23.17
100	4950	0.5	92.45
200	19900	0.25	110.45
200	19900	0.5	152.87
500	124750	0.25	354.23
500	124750	0.5	456.87
750	280875	0.25	643.43
750	280875	0.5	768.89
1000	499500	0.25	1258.12
1000	499500	0.5	1539.15
1500	1125250	0.25	1743.89
1500	1125250	00.5	1934.11
2000	1999000	0.25	2176.51
2000	1999000	0.5	2318.40
3500	6123250	0.25	3245.67
3500	6123250	0.5	3556.44
5000	12497500	0.25	4321.71
5000	12497500	0.5	4568.11

Table 7: Running time (in seconds) of CR-maximal CCES algorithm on Real time datasets

$\|V\|$	$EdgeDensity$	$CR-maximalCCES$
1000	1	1352.56
2000	0.75	2572.90
3000	0.40	3982.12
4000	0.60	5328.49
5000	0.50	6273.81

synthetic data sets comprising 200 to 5000 nodes, and the results are shown in Table 3. A graphical display of the data in Table 3 is shown in Figure 10.

The performance of CR-maximal CCES algorithm is analyzed on sparse, non-sparse and complete graphs. For that, we have generated synthetic data sets comprising 100, 200, 500, 750, 1000, 1500, 2000, 3500, and 5000 nodes each, and the running time is shown in Tables 4, 5, 6. β in Tables 4, 5, 6 denotes edge density of maximal CCES. Figures 11 assures that the running time to detect a maximal CCES for complete input graphs is lower than non-dense input graphs.

CR-values for the graphs with size 1000 nodes or less can be computed in less than 1 hour and graphs with above 1000 are also calculated with reasonable time using the tool UCINET.

In Table 7, A real time chemical database is obtained from open database website

and processed to analyze the efficiency of CR-maximal CCES algorithm. The obtained maximal CCES size of a pair of chemical graphs is 25%. In all the experiments with various size of data sets the proposed CR-maximal CCES algorithm performs efficiently.

7 Conclusion

A maximal CCES is a promising approach for chemical similarity searching and activity predictions in chemical database. The proposed novel idea is that it identifies a node in the tensor product graph with the highest CR-value from where the DFS search is to be started to find at least one maximal CCES, instead of searching from each node of the tensor product graph. The experimental results have proven the efficiency of the proposed method. Further scope for development lies in characterizing graphs with same CR-values. We believe that the size of maximum common edge subgraph (MCES) can be comparatively larger than maximum common connected edge subgraph (MCCES) due to relaxing connectivity constraint. This paper can be extended to find MCCES and MCES using exact and also approximate algorithms.

References

[1] E. Velasquez, E. R. Yera, R. Singh, Determining molecular similarity for drug discovery using the wavelet riemannian metric, in: *BioInformatics and BioEngineering, 2006. BIBE 2006*, IEEE, 2006, pp. 261–268.

[2] R. Gozalbes, F. Barbosa, E. Nicolaï, D. Horvath, N. Froloff, Development and validation of a pharmacophore–based QSAR model for the prediction of CNS activity, *ChemMedChem* **4** (2009) 204–209.

[3] P. Willett, J. M. Barnard, G. M. Downs, Chemical similarity searching, *J. Chem. Inf. Comput. Sci.* **38** (1998) 983–996.

[4] Y. Cao, T. Jiang, T. Girke, A maximum common substructure–based algorithm for searching and predicting drug–like compounds, *Bioinformatics* **24** (2008) i366–i374.

[5] R. J. V. Berlo, W. Winterbach, M. J. D. Groot, A. Bender, P. J. Verheijen, M. J. Reinders, D. D. Ridder, Efficient calculation of compound similarity based on maximum common subgraphs and its application to prediction of gene transcript levels, *Int. J. Bioinf. Res. Appl.* **9** (2013) 407–432.

[6] R. P. Sheridan, S. K. Kearsley, Why do we need so many chemical similarity search methods? *Drug Discovery Today* **7** (2002) 903–911.

[7] H. C. Ehrlich, M. Rarey, Maximum common subgraph isomorphism algorithms and their applications in molecular science: a review, *WIREs Comput. Mol. Sci.* **1** (2011) 68–79.

[8] J. W. Raymond, P. Willett, Maximum common subgraph isomorphism algorithms for the matching of chemical structures, *J. Comput. Aided Mol. Design* **16** (2002) 521–533.

[9] H. G. Barrow, R. M. Burstall, Subgraph isomorphism, matching relational structures and maximal cliques, *Inf. Proces. Lett.* **4** (1976) 83–84.

[10] G. Levi, A note on the derivation of maximal common subgraphs of two directed or undirected graphs, *Calcolo* **9** (1973) 341–352.

[11] M. R. Garey, D. S. Johnson, *Computers and Intractability*, Freeman, New York, 1979.

[12] I. M. Bomze, M. Budinich, P. M. Pardalos, M. Pelillo, The maximum clique problem, in: D. Z. Du, P. M. Pardalos (Eds.), *Handbook of Combinatorial Optimization*, Kluwer, Dordrecht, 1999, pp. 1–74.

[13] U. Brandes, T. Erlebach (Eds.), *Network Analysis: Methodological Foundations*, Springer, 2005.

[14] M. Newman, *Networks: An Introduction*, Oxford Univ. Press, Oxford, 2010.

[15] L. C. Freeman, entrality in social networks conceptual clarification, *Soc. Networks* **1** (1979) 215–239.

[16] J. L. Gross, J. Yellen, *Handbook of Graph Theory*, CRC Press, Boca Raton, 2004.

[17] N. Nikolova, J. Jaworska, Approaches to measure chemical similarity – a review, *QSAR Comb. Sci.* **22** (2003) 1006–1026.

[18] H. Bunke, P. Foggia, C. Guidobaldi, C. Sansone, M. Vento, A comparison of algorithms for maximum common subgraph on randomly connected graphs, in: T. Caelli, A. Amin, R. P. W. Duin, D. de Ridder, M. Kamel (Eds.), *Structural, Syntactic, and Statistical Pattern Recognition*, Springer, Berlin, 2002, pp. 123–132.

[19] D. Conte, P. Foggia, M. Vento, Challenging complexity of maximum common subgraph detection algorithms: A performance analysis of three algorithms on a wide database of graphs, *J. Graph Algorithms Appl.* **11** (2007) 99–143.

[20] J. W. Raymond, E. J. Gardiner, P. Willett, Rascal: Calculation of graph similarity using maximum common edge subgraphs, *Comput. J.* **45** (2002) 631–644.

[21] R. Jain, H. K. Yalamanchili, N. Parekh, Identifying structural repeats in proteins using graph centrality measures, in: A. Abraham, A. Carvalho, F. Herrera, V. Pai (Eds.), *Nature & Biologically Inspired Computing, NaBIC 2009*, IEEE, 2009, pp. 110–115.

[22] G. Abay–Asmerom, R. Hammack, Centers of tensor products of graphs, *Ars Comb.* **74** (2005) 201–212.